Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1989

Real-time implementation of an adaptive depth controller for a submersible.

Williams, James M.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/26201

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

W597802

REAL-TIME IMPLEMENTATION OF AN
ADAPTIVE DEPTH CONTROLLER
FOR A SUBMERSIBLE

by

James M. Williams

December 1989

Thesis Advisor: Roberto Cristi

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4 PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000

8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)		10 SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification) REAL-TIME IMPLEMENTATION OF AN ADAPTIVE DEPTH CONTROLLER FOR A SUBMERSIBLE

12. PERSONAL AUTHOR(S)
WILLIAMS, James M.

13a. TYPE OF REPORT Master's Thesis	13b TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1989 December	15 PAGE COUNT 86

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

17	COSATI CODES		18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	AUV; Adaptive Control; RLS; Variable Structure; Sliding Mode

19 ABSTRACT (Continue on reverse if necessary and identify by block number)
An Autonomous Underwater Vehicle (AUV) is an unmanned submersible vehicle capable of performing a variety of missions. The AUV, which is the subject of this research, is a small prototype vehicle equipped with various control surfaces as well as telemetry devices which provide pertinent measurements of the vehicle states. This research is directed toward the development and implementation of a digital-control program which provides robust depth control of the vehicle. An adaptive parameter estimation routine is used to develop the model of the relationship between the actuator commands and vehicle response. State feedback is then provided using a variable-structure approach. The control algorithm has been implemented through a Turbo Pascal digital-control program executed on a PC/AT computer. Results of the parameter estimation routine and controller implementation are discussed.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL CRISTI, Roberto	22b TELEPHONE (Include Area Code) 408-646-2223	22c OFFICE SYMBOL 62Cx

DD Form 1473, JUN 86 　　　Previous editions are obsolete.　　　SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 　　　UNCLASSIFIED

i

Real-Time Implementation of an Adaptive
Depth Controller for a Submersible

by

James M. Williams
Lieutenant, United States Navy
B.O.E., University of Mississippi, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1989

ABSTRACT

An Autonomous Underwater Vehicle (AUV) is an unmanned submersible vehicle capable of performing a variety of missions. The AUV, which is the subject of this research, is a small prototype vehicle equipped with various control surfaces as well as telemetry devices which provide pertinent measurements of the vehicle states. This research is directed toward the development and implementation of a digital control program which provides robust depth control of the vehicle. An adaptive parameter estimation routine is used to develop the model of the relationship between the actuator commands and vehicle response. State feedback is then provided using a variable structure approach. The control algorithm has been implemented through a Turbo Pascal digital control program executed on a PC/AT computer. Results of the parameter estimation routine and controller implementation are discussed.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Professor Roberto Cristi for his encouragement and patience throughout this project. I would also like to thank Professor Anthony J. Healey for his technical expertise and for the test equipment provided by the Mechanical Engineering Department. Finally, I thank my wife, Tammie, for her support under difficult circumstances.

I. INTRODUCTION

Currently, there is a strong interest within the U.S. Navy concerning the development of an Autonomous Underwater Vehicle (AUV). The possible mission objectives of such a vehicle are far-ranging, including applications in operations such as ASW, and underwater surveillance. The AUV must be capable of following a pre-programmed route to a designated target area, perform its mission and surface for recovery and subsequent data extraction. In response to the Navy's interest in the AUV, a great deal of research has been conducted at NPS, resulting in the construction of a small-scale prototype vehicle.

This work addresses the real-time implementation of a depth controller for the prototype vehicle. In designing a depth controller for the AUV, there are a number of associated problems which are of notable concern. First of all, the equations of motion of the submersible are determined by the nonlinear hydrodynamic forces of the vehicle which are difficult to approximate by conventional means. Also, the dynamics change under varying operating conditions such as speed and vehicle configuration. In addition, the measurement hardware associated with this particular vehicle produce a state measurement which is corrupted with an undesirable time-

varying bias. This time-varying component cannot be modeled, and must be approximated, then removed from the measured signal. The various problems associated with the dynamic modeling of the AUV illuminate the need for an adaptive system identification approach. In this work, the RLS (Recursive Least Squares) algorithm was used to determine the dynamic model of the system. The RLS algorithm accepts measurements of the system input and output and returns the model parameters which, when applied to the model, minimize the error between the estimated and measured system output. This approach provided an accurate representation of the varying dynamics of the system by accounting for the relationship between the input and related output of the system under all operating conditions.

Once the system states and parameters have been estimated, they are applied to a control law in order to provide the next input signal to the system. The controller used in this work is known as a *Variable Structure (VS) controller*. VS controllers have recently been proposed for use in the control of submersibles and other vehicles with nonlinear or unmodeled dynamics [Ref.1]. The variable structure technique implemented in conjunction with an adaptive state estimation algorithm, guarantees robust trajectory control. The remainder of this work addresses the detailed development of the forementioned concepts as well as their physical

implementation. A digital-control program will be developed which provides a diveplane command to the AUV, based on sampled voltage signals representing the states of the vehicle. This thesis is organized to represent the logical progression of the development of the control program from theoretical conception to implementation. Chapter II provides a detailed discussion of the unique dynamics associated with the AUV. Additionally, a linearized model of vehicle dynamics in the vertical plane is developed. Chapter III addresses the design of an adaptive state estimation routine, which provides estimates of the system states needed for feedback. In Chapter IV, the theory supporting the development of the variable structure controller is discussed, and the resulting controller design is tailored for specific application in the AUV depth control system. Chapters V and VI detail the implementation of the developed control concepts through discussion of the hardware configuration and digital program development, respectively. The experimental results of the control process are presented in Chapter VII. The results provided by the parameter estimation routine and controller are segmented to allow for independent analysis. Chapter VIII summarizes the performance analysis of the depth control system, resulting in the statement of conclusions regarding the performance of the developed control process.

II.   AUV DYNAMICS

A.   GENERAL DESCRIPTION OF THE AUV

The AUV considered in this research is 30" in length, with a rectangular body cross section as depicted in Figure 1 [Ref.2]. The vehicle is equipped with twin screws to provide propulsion as well as three sets of control surfaces: rudder, bow planes, and dive planes. The two rudders operate in tandem to control the direction of forward motion of the vehicle. The bow planes ensure that the vehicle maintains a minimum roll angle via symmetrically opposed operation, while the diveplanes control the vehicle's depth. In addition,
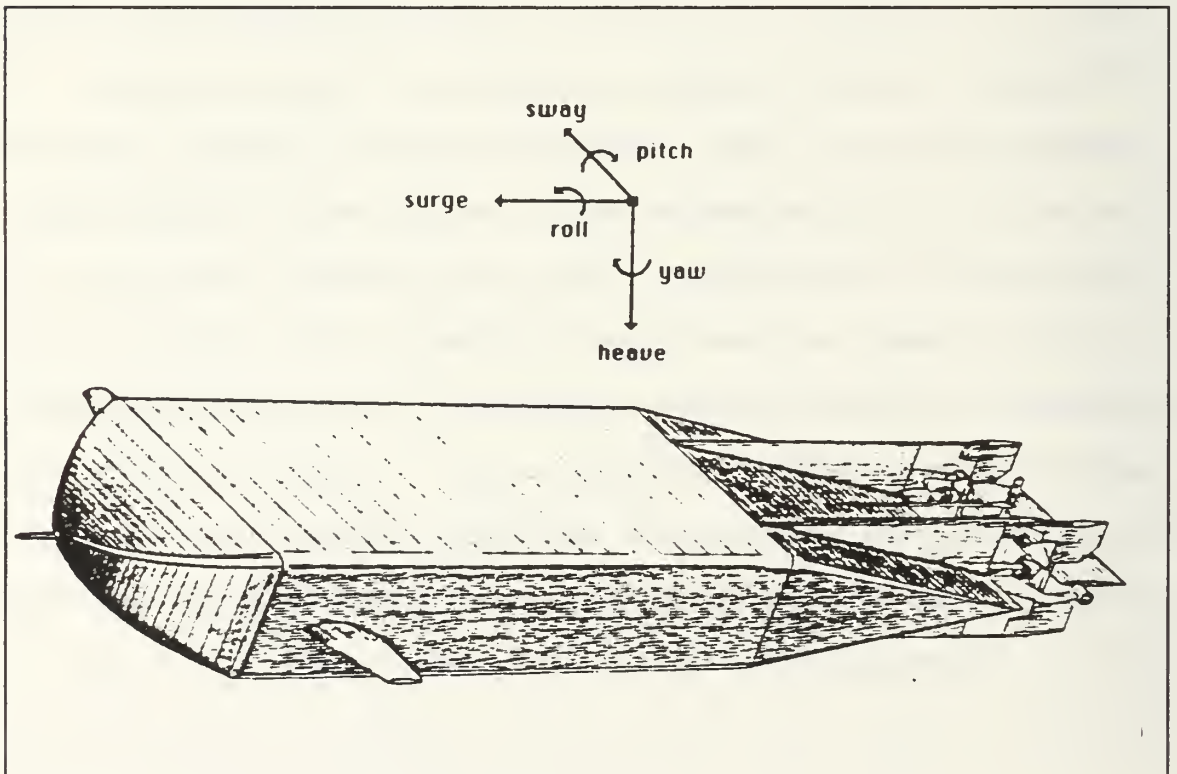
Figure 1   Sketch of the AUV Model

4

the AUV is equipped with a depth cell which provides depth measurements, as well as three rate gyros. The pitchrate, rollrate, and yawrate gyros are positioned within the vehicle in such a way as to provide for measurement of vehicles movement in the pitch, roll, and yaw axis.

B.   DETERMINATION OF THE DEPTH CONTROL MODEL

Mathematical models based on the equations of motion for a body of revolution encompass six degrees of freedom of movement referenced to both the body-fixed coordinate system and the inertial reference frame [Ref.3]. The specific coordinates are represented in Figure 1 and listed below as follows:

u = surge rate $\qquad$ x = surge

v = sway rate $\qquad$ y = sway

w = heave rate $\qquad$ z = heave

p = roll rate $\qquad$ ρ = roll

q = pitch rate $\qquad$ θ = pitch

r = yaw rate $\qquad$ ψ = yaw .

The development of the hydrodynamic model is based on the Navy's SDV-9 Swimmer Delivery Vehicle, and is detailed in Ref. 3 and Ref. 4. In designing a depth-control system, we would

5

intuitively be concerned only with depth, pitch, and pitchrate. Due to the fact that the AUV is a MIMO (Multi-input, Multi-output) system, however, we expect some degree of cross-coupling between the different inputs and outputs of the system. This cross-coupling effect can be seen, for example, when a rudder command is applied to the AUV while operating at a non-zero roll angle. In response to the rudder command, the vehicle would certainly experience a change in depth together with the intended change in direction. In this situation, the depth control system would generate and apply a signal to the diveplanes in order to maintain the desired depth. Schwartz [Ref.3] addressed this cross-coupling and estimated the effect by applying various diveplane, bowplane, and rudder commands to the vehicle while observing the effect of these commands on all system states. For the particular shape of the vehicle considered, analysis of this data led to the conclusion that cross-coupling between the system inputs and outputs is not of significant concern if the bowplane, diveplane, and rudder are operated independently. It is by this assumption of independence that we can model the dynamics in the vertical plane as a SISO (Single-input, Single-output) system.

C.   DERIVATION OF THE MATHEMATICAL MODEL

Detailed mathematical models of submersibles are available from many sources. Basically, they are derived from laws of

hydrodynamics and lead to very complex sets of nonlinear differential equations. A simpler class of models can be obtained by observing the behavior of the state vector under several command inputs. In a diving maneuver, the system states of importance are pitchrate (Q), pitch (θ), and depth (Z), while the command is the diveplane angle (δ). Observing the behavior of the vehicle under consideration, the dynamics between the diveplane angle (δ), the pitchrate (Q), and pitch (θ) are approximated as

$$\dot{Q}(t) = aQ(t) + b\delta(t) \tag{1}$$

$$\dot{\theta}(t) = Q(t) \tag{2}$$

while the model for depth (Z) becomes,

$$\dot{Z}(t) = -V\sin(\theta(t)) \tag{3}$$

with V being the forward velocity of the vehicle. Justification for (3) can be seen by examining the proportionality between the velocity of the vehicle and the rate of change of depth. The velocity of the vehicle can be expressed as a vector of known magnitude, directed at an angle θ, relative to the horizontal plane. By simple vector analysis, the velocity vector can be resolved into both horizontal and vertical components for a given θ and velocity.

The resolvant vertical component of the velocity vector is then taken to be the rate of change in depth as in (3).

The parameters a and b are dependent on several factors, such as operating conditions and velocity. To give a general idea of the magnitude of these parameters for small pitch angles, (1) through (3) are written in state space form as,

$$\begin{bmatrix} \dot{Q} \\ \dot{\theta} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -1.8 & 0 & 0 \\ 1.0 & 0 & 0 \\ 0 & -2.1 & 0 \end{bmatrix} \begin{bmatrix} Q \\ \theta \\ Z \end{bmatrix} + \begin{bmatrix} 2.531 \\ 0 \\ 0 \end{bmatrix} \delta \qquad (4)$$

with Q in radians/second, θ in radians, and Z in feet. This model applies only to the specific vehicle under study, cruising at a velocity of 2.1 feet/second. If the basic shape, mass, or velocity of the vehicle changes, the result will be a corresponding change in the parameters which represent the dynamic model of the system. For this reason, the conventional approach to modeling the system dynamics is generally considered unsatisfactory. The parameter estimation routine implemented in this work is an adaptive scheme which accounts for the changing dynamics of the system. This adaptive parameter estimation approach is discussed in detail in the following chapter.

III.   ESTIMATOR DESIGN

A.   BACKGROUND

A state estimator, or observer, is one of the fundamental
elements of any practical feedback-control system. In order
to stabilize the dynamics of a system, the system states must
be available for measurement or otherwise be estimated by a
suitable algorithm. In designing the AUV, certain limitations
were placed on the selection of sensor hardware which made the
direct measurement of all system states impossible. Ideally,
the AUV would be equipped with a position gyro to measure the
vehicle pitch angle, as well as a pitchrate gyro and depth
cell to measure vehicle pitchrate and depth. Unfortunately,
the inclusion of a pitch gyro was impractical due to both the
unit cost and space requirements for installation. The
quality of the pitchrate gyro selected was also limited by
unit cost. The pitchrate gyro selected was a low quality unit
typically found in hobby shops. As a consequence, the
pitchrate gyro produces a sensor signal corrupted by a slowly
varying DC bias voltage. In summary, due to the physical
limitations of the sensors used, depth is the only state which
is accurately measureable.

B.   GENERAL DESIGN APPROACH

To generate estimates of the remaining two states,
required the design of a state estimator which accepts depth
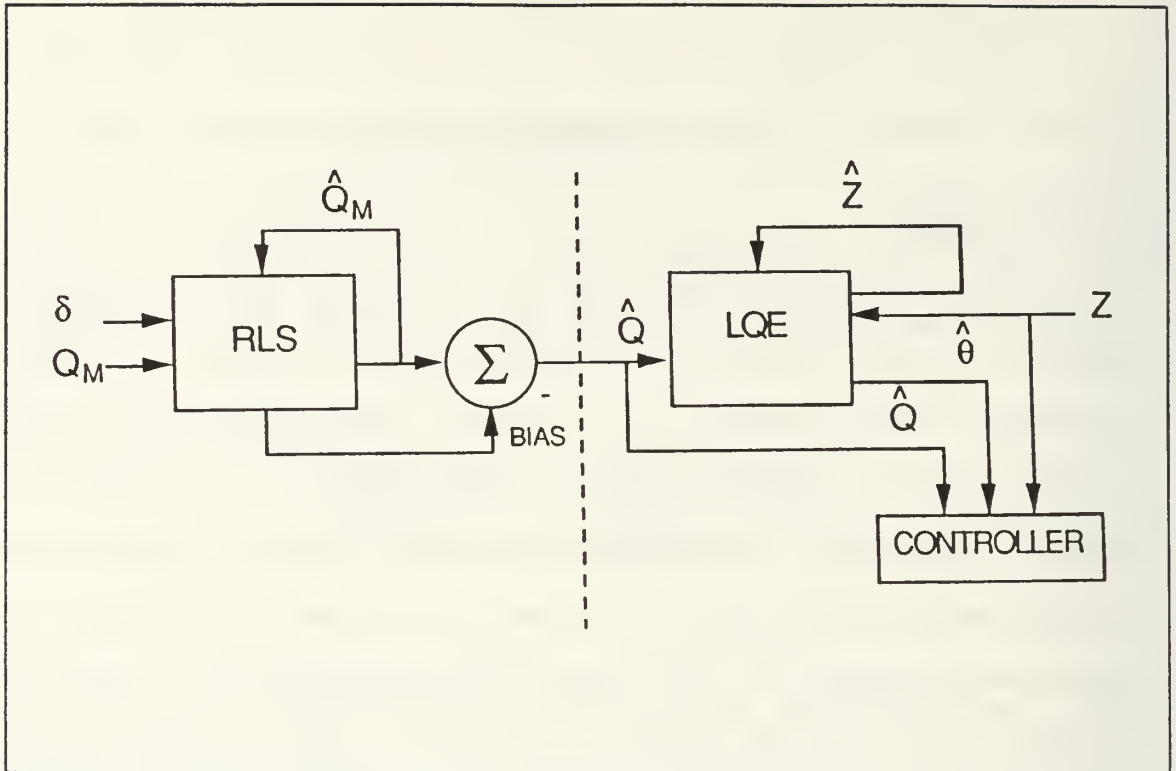
Figure 2   Estimator Design Problem Division

and corrupted pitchrate measurements as inputs and generates
accurate estimates of all system states at the output.   The
general approach, depicted in Figure 2, is to use a hybrid
design encompassing both RLS (Recursive Least Squares) methods
to estimate the bias embedded in the pitchrate gyro signal,
as well as LQE (Linear Quadratic Estimator) techniques to
provide a pitch estimate given an estimate of pitchrate and
the measured depth of the vehicle.

C.   OVERVIEW OF THE RLS ALGORITHM

Consider a first-order SISO system with unknown dynamics.
Let the input sequence to the system be $u(t)$ and the system
output be $y(t)$.   The plant of such a system can be modeled by

a first order difference equation:

$$y(t) = ay(t-1) + bu(t-1) \tag{5}$$

$$y(t) = \begin{bmatrix} y(t-1) & u(t-1) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{6}$$

$$y(t) = \phi^T(t-1)\underline{\theta} \tag{7}$$

where ϕ^T is the regression vector containing the history of the system inputs and corresponding outputs, and $\underline{\theta}$ is the parameter vector containing the system parameters a and b. Since the dynamic model of the system is fully describable by the parameters a and b, our objective is to estimate the parameter vector $\underline{\theta}$ from the available input and output data. In order to accomplish this, the system is first modeled in state space form as,

$$\underline{\theta}_{t+1} = \underline{\theta}_t + \omega_t \tag{8}$$

$$y_t = H_t \underline{\theta}_t + v_t \tag{9}$$

where H^T is the regression vector ϕ^T(t-1). The term ω_t in (8) accounts for any drifts in the parameters which occur through time, and v_t is the Gaussian measurement noise. Let us define the autocovariance of ω_t and v_t respectively.

$$S_t = E\left[\omega_t \omega_t^T \right] \tag{10}$$

11

$$R_t = E\left[v_t v_t^T \right] \tag{11}$$

The next step is to derive a predictor equation based on the system model. The vector $\underline{\theta}$ is known to contain the constant parameters a and b. Therefore, the predictor equation for $\underline{\theta}$ is,

$$\hat{\underline{\theta}}_t = \hat{\underline{\theta}}_{t-1} . \tag{12}$$

Accordingly, the predicted output of the system is determined by the product of the transition matrix H^T and the predicted $\underline{\theta}$ vector.

$$\hat{y}_t = H_t \hat{\underline{\theta}}_{t-1} \tag{13}$$

In this particular framework, the Kalman filter algorithm is a least−squares approach to state estimation. The predicted output of the system is computed using (13), and then compared to the actual measured output to determine the prediction error. The Kalman filter serves to minimize the variance of this prediction error through repeated applications of the corrector equation.

$$\hat{\underline{\theta}}_t = \hat{\underline{\theta}}_{t-1} + K_t \left(y_t - \hat{y}_t \right) \tag{14}$$

where,

$$\hat{\underline{\theta}}_t = E\left[\underline{\theta}_t | y_{t-1}, y_{t-2}, \ldots, y_0 \right] . \tag{15}$$

The Kalman gain vector K_t is dependent upon the reliability of the estimation and is determined as,

$$K_t = \frac{P_{t-1}\ \phi_t}{R_t + \phi_t^T P_{t-1}\ \phi_t}$$ (16)

where, for a system of order n, K_t is a (n × 1) vector and P_t is the (n × n) matrix representing the accuracy of the current parameter estimation. As the number of recursions increases, the magnitude of the prediction error decreases and the values of the P_t matrix become correspondingly smaller. The obvious result being that as the prediction becomes more accurate, the system model equations more closely approximate actual system response, resulting in an improved system model. The estimation error covariance matrix is computed, then updated after each iteration using the following formula [Ref. 5].

$$P_t = P_{t-1} - \frac{P_{t-1}\ \phi_t\ \phi_t^T P_{t-1}}{R_t + \phi_t^T\ P_{t-1}\ \phi_t} + S_t$$ (17)

The initial condition P(0) is the covariance of the error in the initial state, and represents the a-priori information available concerning the estimated parameters.

D. PITCHRATE BIAS ESTIMATION

Application of the RLS algorithm to the particular case of estimating a signal with an embedded bias proved to be only slightly different than that of the general case. Let us

13
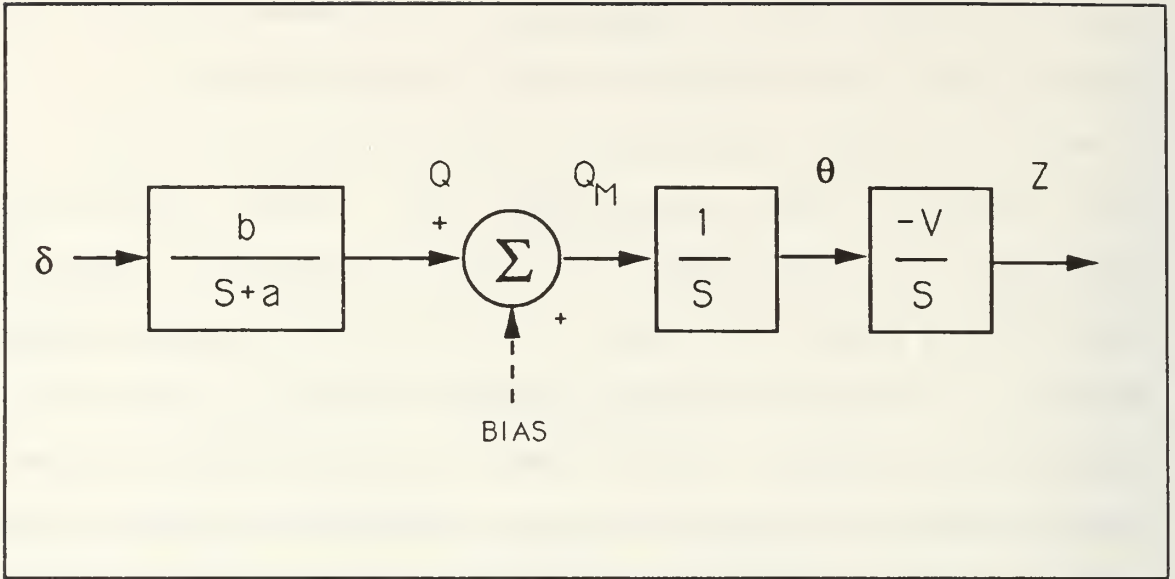
Figure 3   AUV Dive System Representation

consider the ARMA (Auto-Regressive Moving Average) model between the diveplane input and pitchrate, as shown in the system representation of Figure 3. Note that the system cannot be modeled by (5) in its present form due to the added bias. Through application of the principles of the RLS algorithm; however, the bias can be accurately estimated. To accomplish this, the ARMA model of the difference equation from the diveplane input to the pitchrate output is introduced as it would be with no bias present,

$$Q(t) = \alpha Q\,(t-1) + \eta \delta(t-1) \tag{18}$$

where α and η are the discrete equivalents of the continuous

14

time system parameters a and b. If we let Q(t)=Q$_m$(t)-ß(t); after some manipulation, the dynamic model in terms of Q$_m$ becomes,

$$Q_m(t) = \alpha Q_m(t-1) + \eta \delta(t-1) + \left(\beta(t) - \alpha\beta(t-1)\right) \tag{19}$$

where ß is the is the discrete-time bias term. Since ß is piecewise constant over a small interval, (19) may be further simplified by grouping all terms which are a function of ß and replacing them with a single constant term γ, as follows.

$$Q_m(t) = \alpha Q_m(t-1) + \eta \delta(t-1) + \gamma \tag{20}$$

With this final simplification complete, the system model from the diveplane input to the bias corrupted pitchrate output Q$_m$ is expressed as,

$$Q_m(t) = \begin{bmatrix} Q_m(t-1) & \delta(t-1) & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \eta \\ \gamma \end{bmatrix} \tag{21}$$

Note that the $\underline{\theta}$ vector does not contain the bias term ß, that we wish to estimate. The $\underline{\theta}$ vector, however, does contain the parameter γ identified earlier as,

$$\gamma = \beta(t) - \alpha\beta(t-1) \tag{22}$$

From the estimate of γ, the discrete bias term ß must be approximated, so that it can be later subtracted from the estimated corrupted pitchrate measurement. To explain this

15

process, let us recall that this approach to bias estimation was based on the fact that ß could be approximated as a constant term over a small time interval. With this assumption in mind, we can approximate that ß(t-1) is approximately equivalent to ß(t), simplifying the relationship to

$$\gamma = (1-\alpha)\beta(t-1) \tag{23}$$

where γ and α are estimated components of $\underline{\theta}$. The discrete bias term ß may now be expressed as a function of γ and α;

$$\beta(t) = \frac{\gamma}{(1-\alpha)} \tag{24}$$

The RLS algorithm can now be applied in this case exactly as demonstrated earlier for the generic case. After each iteration of the algorithm, a current estimate of the parameter vector $\underline{\theta}$ is generated. Using the estimate of $\underline{\theta}$, the corrupted pitchrate and bias term are estimated. The bias is then subtracted from the corrupted pitchrate estimate to yield a pure pitchrate estimate. [Ref. 5]

E.   LINEAR QUADRATIC ESTIMATOR DESIGN

The estimator design, to this point, provides only an estimate of the pitchrate by removing the bias. In order to control the depth of the AUV, all states must be available for feedback. To generate estimates of vehicle pitch, let us consider the reduced system depicted in Figure 4. The

16
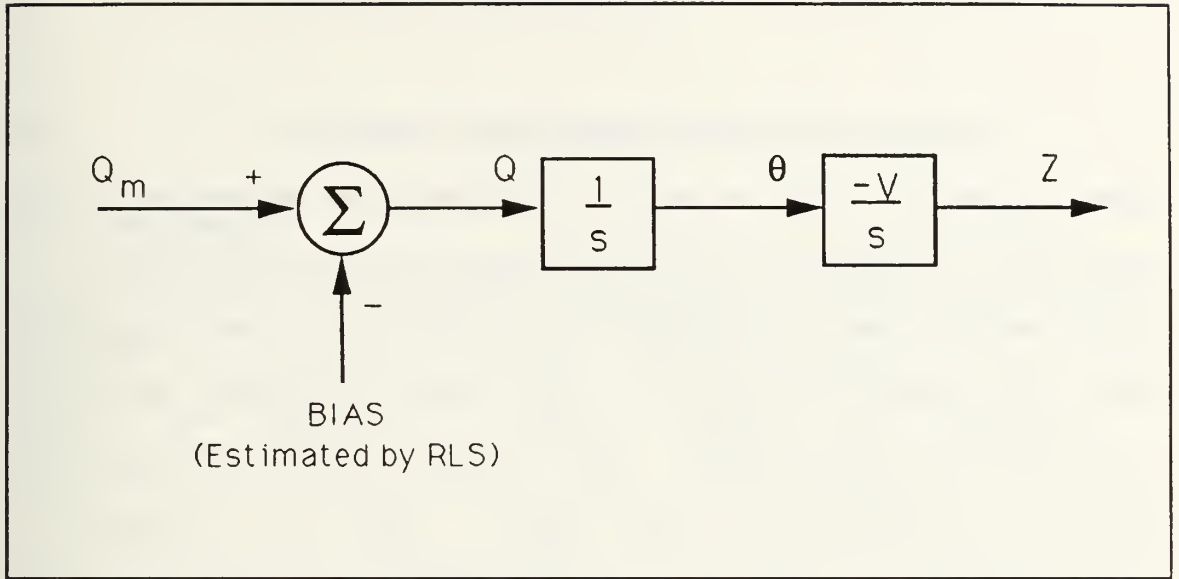
Figure 4   Second Order System Model

pitchrate signal Q, which is the output of the RLS estimator,
is treated as an input to the second order system relating
pitchrate to depth. Throughout the design of the LQE we need
only to be concerned with this portion of the system.

To begin the analysis, the reduced system is represented
in the form,

$$\begin{bmatrix} \dot{\theta} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -2.1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ Z \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} Q + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \omega_t \qquad (25)$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ Z \end{bmatrix} + v_t \qquad (26)$$

where ω_t is the process noise and v_t is the measurement noise,
each with a zero mean value.   Furthermore, let the noise
covariances be expressed as in (27) and (28)   [Ref.6];

$$E\left[\omega_t \omega_t^T\right] = S_t \tag{27}$$

$$E\left[v_t v_t^T\right] = R_t \tag{28}$$

The values of the noise covariances are selected by the designer to maximize the performance of the estimator. In this design, several combinations of S_t and R_t were specified and the resulting Kalman gains were computed using the software package MATLAB. The various sets of Kalman gains were then applied to the LQE in order to evaluate the performance of the estimator. The Kalman gains which provided the best performance were then implemented in the digital program through use of the equation,

$$\dot{\hat{\underline{X}}} = A\hat{\underline{X}} + Bu + \underline{K}\left(y - c\hat{\underline{X}}\right) \tag{29}$$

where the estimation error is,

$$y - C\hat{X}. \tag{30}$$

In the following chapter, the estimated states provided by the state estimator will be used in a nonlinear feedback control configuration to generate the closed loop input to the system.

IV.   VARIABLE STRUCTURE CONTROLLER DESIGN

A.   BACKGROUND

The dynamics of autonomous underwater vehicles are complex and highly nonlinear by nature. The hydrodynamic forces which govern the behavior of such a system are dependent on parameters such as vehicle speed, acceleration and inertia. In addition, there are a variety of unmeasurable disturbances which add to the uncertainty of the vehicle response. In order to design a robust controller using linear control techniques, it would be necessary to devise several system models and associated control schemes in order to represent the system under the varying operating conditions. For this reason, modern linear-control techniques generally prove to be too complex and computationally intensive to provide adequate control of underwater vehicles.

One recently developed control methodology which deals directly with the control of nonlinear systems is the *variable structure controller*. Using the variable structure approach, robust trajectory control is guaranteed despite the presence of unmodeled or time-varying system dynamics. The remainder of this chapter deals with the development and implementation of such a controller.

B. VARIABLE STRUCTURE CONTROLLER DEVELOPMENT

Throughout the development of the variable structure controller, the system to be controlled is expressed in state space form as,

$$\underline{\dot{X}} = A\underline{X} + B\left(\delta_0 + \Delta f(X,\delta_0)\right) \tag{31}$$

where the uncertainty of the system model is expressed as the quantity $\Delta f(x,\delta_o)$. This uncertainty can arise from a variety of factors such as the presence of unmodeled dynamics in the system. The origin of this uncertainty is of little concern in the development of the variable structure controller. In the development of the variable structure controller, we assume knowledge of an upper bound of the disturbance term $F(x,\delta_o)$ as,

$$\left\|\Delta f(X,\delta_0)\right\| \leq F(X,\delta_0) \tag{32}$$

with $F(x,\delta_o)$ known for all x and δ_o. On the basis of (31) and the known bound on Δf, a controller can be designed to drive the state vector $\underline{x}(t)$ to within a finite bound around zero. In order to accomplish this, let $\underline{C}^T$ be a left eigenvector of the matrix A, corresponding to a marginally stable eigenvalue λ such that

$$\underline{C}^T A = \lambda \underline{C}^T \tag{33}$$

20

By multiplying both sides of (31) by $\underline{C}^T$ and substituting (33), we obtain

$$\underline{C}^T \underline{\dot{X}} = \lambda \underline{C}^T \underline{X} + \underline{C}^T B \left(\delta_0 + \Delta f(X, \delta_0) \right). \tag{34}$$

Combining terms in (34) results in the assignment of the new variables

$$\sigma(t) = \underline{C}^T \underline{X}(t) \tag{35}$$

$$\gamma = \underline{C}^T B. \tag{36}$$

Additionally, the feedback law selected to generate the closed loop diveplane command is

$$\delta_0 = -K(x) \operatorname{Sign}(\sigma(t)) \tag{37}$$

where K(x) is a positive scalar function, such that $K(x) \geq F(x)$. Making the substitutions of (35) and (36) into (34), yields

$$\dot{\sigma}(t) = \lambda \sigma(t) + \gamma \left(\delta_0 + \Delta f(X, \delta_0) \right) \tag{38}$$

where the diveplane command, δ_0 is determined by (37). With the system expressed in the form of (38), it can be proven that

$$\lim_{t \to \infty} \sigma(t) = 0 \tag{39}$$

21

where $\sigma(t)$ represents a linear combination of system states. [Ref. 1] Note in (39), that $\sigma(t)$ will approach zero within a small bound, as determined by the magnitude of Δf.

Examining the development of the controller at this stage, we can demonstrate why this method is generally referred to as the *sliding mode* approach. First, note that (39) implies that the state vector $\underline{x}(t)$ tends toward the surface

$$\underline{C}^T \underline{X}(t) = 0 \tag{40}$$

which is a hyperplane in the n-dimensional space of the state. The plane of the surface $\sigma(t)$ becomes a switching line as shown in Figure 5, and the control law, (37), assumes different polarities according to which half plane the state is in. Once the state is on the switching line, it remains there by virtue of (39). By properly selecting the vector $\underline{C}$ we obtain

$$\underset{t \to \infty}{Lim} \underline{X}(t) = 0 \,. \tag{41}$$

To prove this claim, we will examine the derivative of the quantity σ^2 which is indicative of the slope of $\sigma(t)$ vs. time.

$$\frac{1}{2}\frac{d}{dt}\left(\sigma^2(t)\right) = \sigma(t)\dot{\sigma}(t) = \sigma(t)\left[\lambda\sigma(t) + \gamma\delta_0 + \gamma\Delta f\right] \tag{42}$$

22
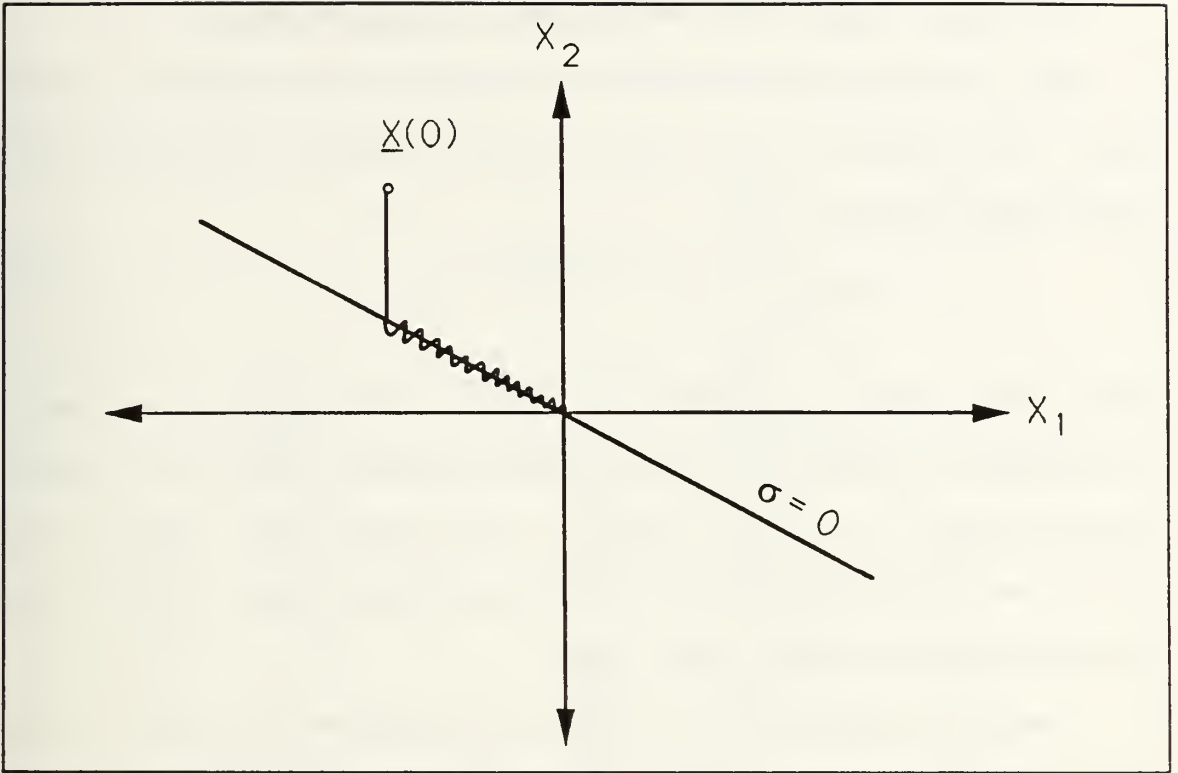
Figure 5   Representation of Switching Line Behavior

Applying the feedback law, (37), and imposing the conditions $\lambda = 0$ (marginally stable) and $K(x) \geq \|\Delta f\|$ for all x, we obtain

$$\sigma(t)\dot\sigma(t) = \sigma(t)\left(-K(x)\,\text{Sign}\left(\delta(t) + \Delta f\right)\right) \qquad (43)$$

$$\sigma(t)\dot\sigma(t) \leq -K(x)|\sigma(t)| + \sigma(t)\Delta f. \qquad (44)$$

Examining (44) in light of the imposed conditions, it is apparent that the slope of $\sigma^2(t)$ vs. time is always negative. The most important result of this conclusion is that as the states of the system track along the sliding surface, they will ultimately converge to zero regardless of which half-plane the system is operating in. [Ref. 7]

23

C. VARIABLE STRUCTURE CONTROLLER IMPLEMENTATION

The controller developed in the previous section assumes that the nominal linear model of the system (not accounting for disturbances) is

$$\dot{\underline{X}} = A\underline{X} + Bu \qquad (45)$$

where the system is marginally stable. Placing the constraint of marginal stability on the system ensures that the sliding surface $\underline{c}^T\underline{x} \to 0$, with $\underline{c}^T$ being a left eigenvector of A. From the previous discussion, the states of the system, $\underline{x}(t)$, will follow the switching line to 0.

Before applying the variable structure technique to the AUV, the system model must be partially compensated as shown in Figure 6. To accomplish this, an additional feedback loop is implemented to stabilize the dynamics from diveplane command to pitch. The value of the feedback coefficient was selected to provide two eigenvalues at -0.9 and the remaining eigenvalue at the origin, making the system marginally stable. Additionally, the depth error (rather than depth) is included in the state vector so that the state vector becomes,

$$\underline{X} = \begin{bmatrix} Q \\ \theta \\ Z_d - Z \end{bmatrix} = \begin{bmatrix} Q \\ \theta \\ e \end{bmatrix} \qquad (46)$$

where Z_d is the ordered vehicle depth.

24
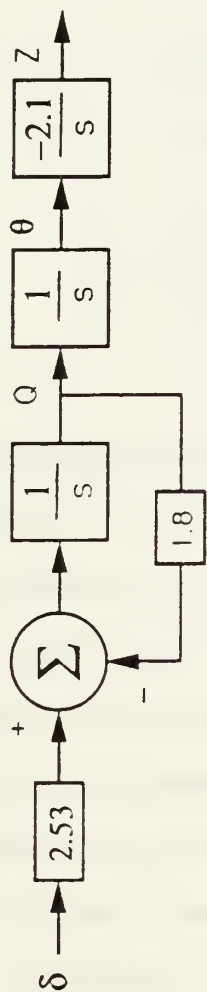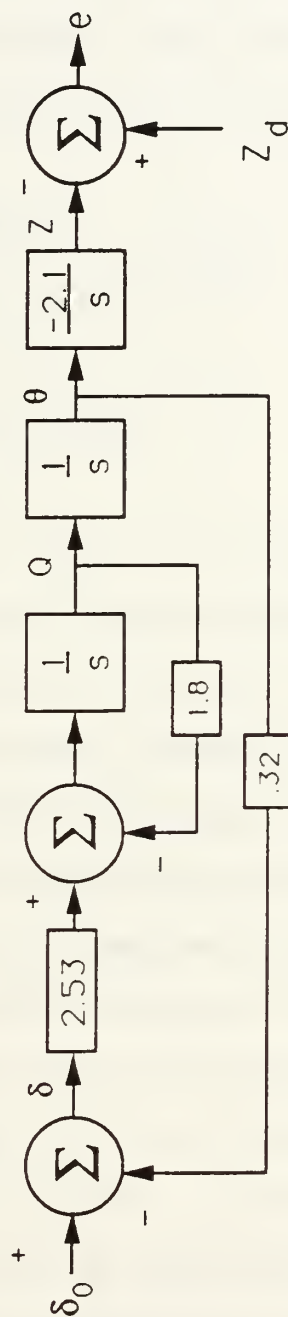
Figure 6   Diagram of Original and Partially Compensated System

25

Using the partially compensated model, the variable structure controller implementation can be applied to the system as previously discussed. Using the MATLAB software package, the left eigenvector of the system, corresponding to the matrix A of (45) and the eigenvalue $\lambda = 0$, is computed as

$$\underline{C}^T = \begin{bmatrix} 0.5556 & 1.0 & 0.2143 \end{bmatrix} \tag{47}$$

leading to the definition of the signal $\sigma(t)$,

$$\sigma(t) = \begin{bmatrix} 0.5556 & 1.0 & 0.2143 \end{bmatrix} \begin{bmatrix} Q(t-1) \\ \theta(t-1) \\ e(t-1) \end{bmatrix} \tag{48}$$

corresponding to the sliding surface. One disadvantage associated with using the signum function to generate the diveplane command δ_o is that the controller exhibits excessive diveplane chatter once the vehicle has reached its steady-state operating depth. To eliminate this problem, the satsign function as illustrated in Figure 7 can be used in place of the signum function. As a result, when the vehicle is near its ordered depth and both pitch and pitchrate are approaching 0, the controller will enter the linear region of operation ranging from $-\Delta\sigma$ to $\Delta\sigma$, where $\Delta\sigma$ is a parameter specified by the designer. Outside this linear region, the diveplane command saturates as it would using the signum function. A large negative value of σ will result in a full-scale positive
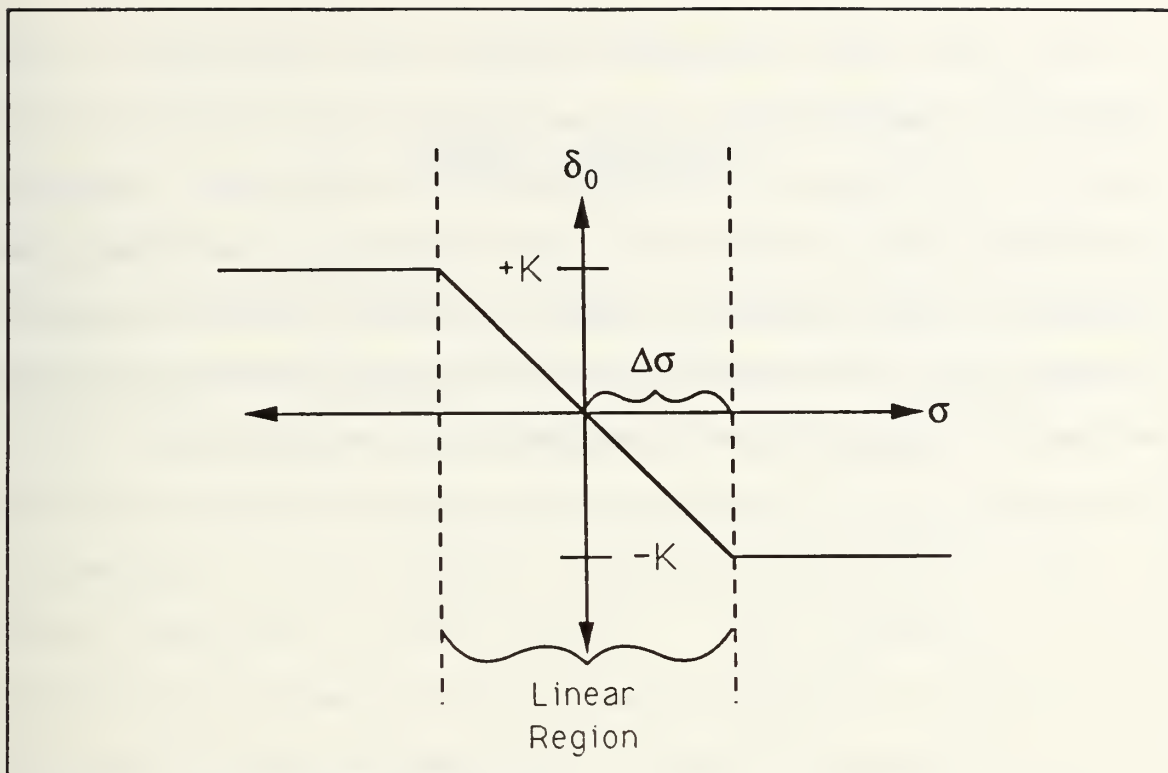
Figure 7   Satsignum Function

diveplane command being asserted with the effect of driving σ into the linear region of control and ultimately to 0. The value of the diveplane command of the original system is derived from the augmented diveplane command, δ_o, through use of the equation,

$$\delta = \delta_0 - 0.32\theta \qquad (49)$$

The feedback coefficient of 0.32 in (49) was selected to ensure that the poles of the partially compensated system were placed at -0.9.

27

The only task remaining before the controller can be physically implemented is the selection of the variables $K(x)$ and $\Delta\sigma$ in Figure 7, which greatly affect the response characteristics of the AUV. These parameters were selected through a trial-and-error approach during the testing phase of this work. The process used and conclusions regarding the selected values will be discussed in the Chapter VII.

V.   ANALOG SIMULATION

A.   GENERAL

As previously discussed, the AUV under study is equipped with a pitchrate gyro and depth cell to provide measurements of the modeled states of the system. The underlying objective of this research has been to use telemetry data from the AUV to generate a state feedback command in order to control the depth trajectory of the vehicle. All procedures to this point have been based on the assumption that the required data is available for processing by the control program. Contrary to this assumption, it was discovered during testing that the pitchrate gyro had become inoperative. Repeated attempts were made to correct the problem with no results.

With no other options available, the decision was made to simulate the AUV in a diving operation using an analog computer. In order to simulate the AUV dynamics on an analog computer, the linearized differential equations describing the system have been implemented through the use of integrators, amplifiers, and attenuating potentiometers. Although this method produces a reasonably accurate simulation of the vehicle dynamic response, the nonlinear components of vehicle motion, such as changes in speed during maneuvering, cannot be accounted for. On the other hand, there are many substantial advantages associated with the use of an analog

29

simulation. Since all system states are generated by the analog computer, they are also available for comparison with state estimates generated within the control program. In addition, the pitchrate bias can be injected into the simulation model as a known quantity rather than as an unknown disturbance, allowing for accurate performance analysis of the RLS algorithm.

B. DESCRIPTION OF THE ANALOG SIMULATION

The analog computer used consisted of an array of integrators, amplifiers, and potentiometers as well as a multi-position voltmeter which is capable of displaying voltage measurements at various system nodes. All amplifiers and integrators operate within a voltage range of ±10 volts. Integrators must be used in conjunction with an amplifier which is capable of providing amplifications of 0.1, 1.0, and 10.0. Negative feedback between the integrator and its associated amplifier provides stabilization. Amplifiers can also be used independently when configured to operate as either a summing junctions or inverters as required. The potentiometers are used as attenuators and provide amplification of their inputs only when used in series with an amplifier. [Ref. 8]

In developing the analog simulation of the AUV, the linearized equations of motion were programmed into the analog computer as depicted in Figure 8. The amplification applied

to each integrator has been determined based on the desired conversion factor from volts to physical units for the system states.   To explain how these scaling factors were selected, let us begin our analysis at the system output.   With the maximum output of each amplifier being 10 volts, the corresponding depth scale of 1 volt equals 1 foot is assigned, allowing for a maximum model depth of 10 feet.   Moving backwards through the system diagram we divide the unit conversion factor by the amplification factor of each amplifier and multiply by 10 for each attenuating potentiometer.   This procedure is repeated to determine the conversion from volts to physical units for each state as well as the diveplane input.   The voltage values of the system states are sampled at a frequency of 20 Hz by the A/D (Analog to Digital) converter.   The previously determined conversion from volts to physical units is applied to these quantities by the digital control program in order to determine the appropriate physical quantity in radians, radians/sec, etc. Likewise, the diveplane command is converted to volt units prior to being applied to the D/A (Digital-to-Analog) converter.
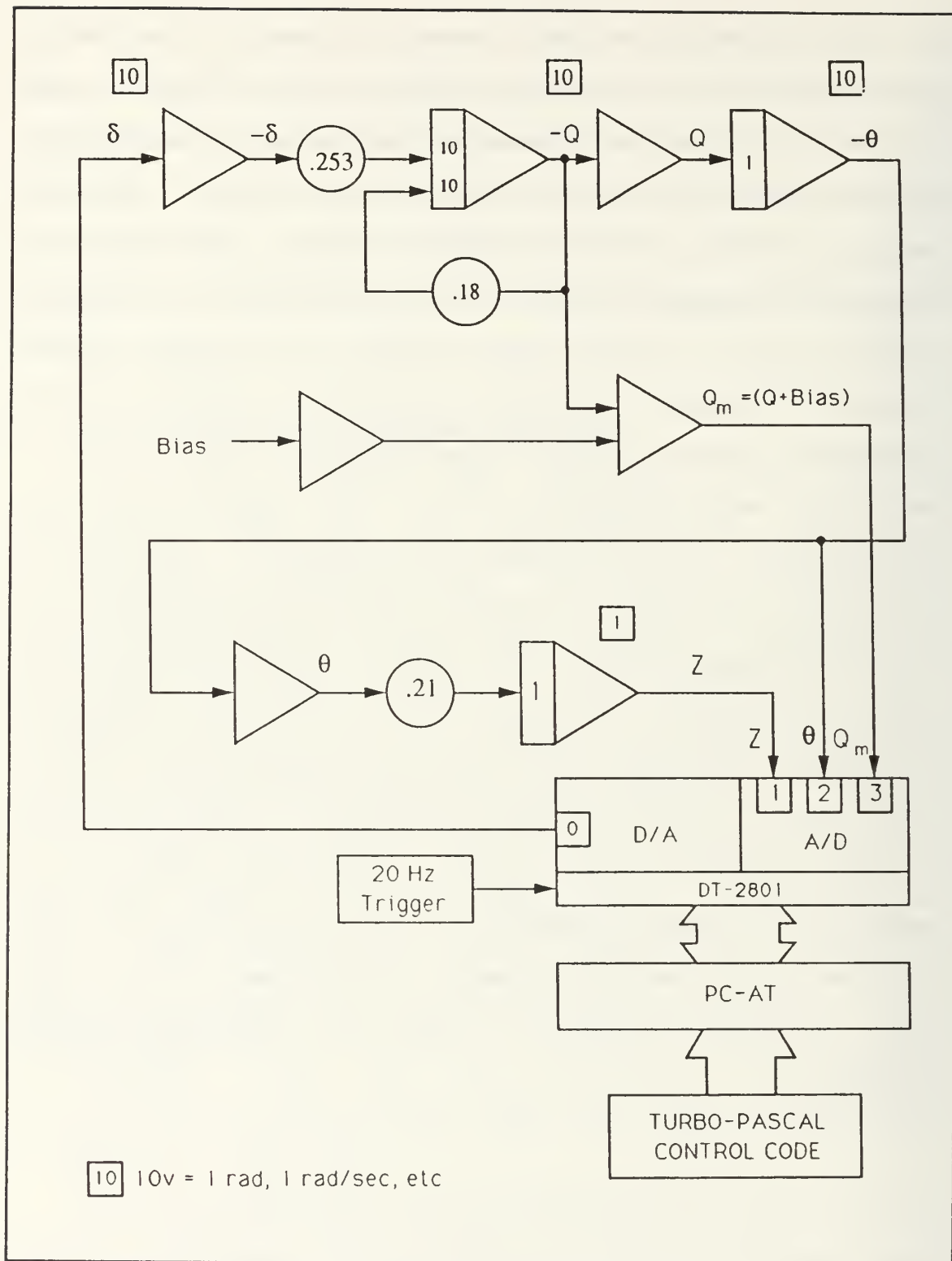
Figure 8 Schematic of the Analog Simulation

32

VI.   DIGITAL PROGRAM IMPLEMENTATION

A.   GENERAL

Thus far, the basic principles and theories upon which the control system design is based have been addressed. The model of the AUV dynamics has been presented and related nonlinearities, as well as model uncertainties, have been discussed. Additionally, a state estimation methodology was identified which addresses the unique parameter identification problems associated with the AUV. Finally, a controller design has been developed which guarantees robustness in the presence of unmodeled nonlinearities in the AUV dynamics. Implementation of the design methodology requires a digital algorithm which accepts the state measurements and desired vehicle depth as inputs and generates the correct diveplane command to be applied to the vehicle.

The programming language selected for use in implementing the digital program was Turbo Pascal version 3.0. Although more powerful languages are available for this application, Turbo Pascal has been selected due to its compatibility with existing hardware used in the implementation. The digital autopilot program is contained in the Appendix.

B. DIGITAL/ANALOG INTERFACE

Before addressing the development of the digital control program, it is necessary to discuss the inter-relationship between the digital controller and the analog simulator. Control systems such as the one developed are termed *sampled data control systems*. This distinction arises from the fact that a digital controller is used to control a system which is operating in continuous time. In the specific case of the AUV, the system states are sampled at a predetermined rate with their respective values held constant between sample intervals. The digital control program must then access these sampled values and process them in order to determine the correct diveplane command to be applied. This interface between the digital computer and the AUV is accomplished through the utilization of both hardware and software.

The data acquisition is implemented using the DT-2801 data translation board. The DT-2801 consists of 3 A/D channels and 1 D/A channel which are accessed by reading data from or writing data to a specified register. The sampling frequency of the DT-2801 is controlled by an external trigger input operating at 20 Hz. The three A/D channels sample and digitize the measured system values and stores the binary equivalent values in an output register. The D/A channel reads the digitized diveplane command from the specified input register, converts the binary value to its voltage equivalent

and applies the command to the rudders. The DT-2801 board is controlled by its associated software routine, PCLAB.

Since the DT-2801 registers only accepts binary values as inputs to its registers, a suitable algorithm must be developed to convert binary to decimal for data entering the program and decimal to binary for data applied to the D/A input registers. Conversion of a physical value to a binary value, or vice versa, depends on the number of bits of resolution used by the converter. The DT-2801 is a 12-bit converter and can, therefore, support 2^{12} or 4096 Number of Codes (NOC) [Ref. 9]. The maximum bipolar voltage range of the vehicle simulator is from ±10 volts. Using this configuration, a voltage of -10 volts corresponds to a NOC of 0, and a voltage of 10 volts corresponds to a NOC of 4096. The algorithm used to convert between digital and decimal values is contained in Ref. 9, PCLAB Users Manual. The implementation of the conversion algorithms is contained in procedures "GET DIGITAL SENSORY DATA" and "GENERATE DIVEPLANE COMMAND" in the digital control program.

C.   DIGITAL PROGRAM DEVELOPMENT

The digital control program represents the innermost functional level of the digital autopilot echelon. All other components of the AUV controller serve only to provide information to, or apply information from, the digital control program. The digital program consists of several integrated

modules called procedures, which function as either utilities or provide application specific functions. Fortunately, past research contained in Ref. 10, has resulted in the development of all utility procedures encompassing data management, user interface, as well as program flow. These various procedures do not address the specific application of the controller under development and are not discussed in this paper. The focus of the remainder of this chapter will, therefore, be the development of the procedures "EST" and "GENERATE DIVEPLANE COMMAND", which address the implementation of the specific state estimator and controller designs.

1. Procedure EST

The procedure EST provides an estimate of the system states using a combination of the RLS algorithm and LQE as discussed in Chapter II. The main body of EST predicts and removes the bias from the pitchrate signal and applies the pure pitchrate signal at the input of the LQE. To accomplish this, EST employs a number of subordinate procedures, each providing a specific function. The following paragraphs describe the structure and use of each of the subordinate procedures.

a. Procedure INITIALIZE ARRAYS

This procedure initializes all system variables on the first call to the procedure EST. This is accomplished by using a count variable in the main program. At the first iteration of the program, the count variable is set to 1

before the procedure EST is called.   The program branches to INITIALIZE ARRAYS, and the count is incremented upon return to EST.

    b.   **Procedure FILTER**

        This procedure provides lowpass filtering for both the measured pitchrate and the applied diveplane command before entering the RLS algorithm.   Filtering the input and output data reduces the high frequency components of the signals which results in a greater speed of convergence of the RLS algorithm.   Filtering both the input and output data has no effect on the accuracy of parameter estimation.

    c.   **Procedure UPDATE**

        UPDATE is used to form the new regression vector of input and output data.   As the newest sample values are read, they are placed in the vector ϕ, and the current values are deleted.   This vector ϕ is later used in predicting the next set of outputs from the RLS algorithm.

    d.   **Procedure KGAIN**

        This procedure calculates the Kalman gain matrix which is to be applied in order to minimize the variance of the estimation error.   In order to accomplish this, KGAIN accepts as inputs, the regression vector ϕ, the past error covariance matrix P, and the previously calculated Kalman gain matrix K.   In the process, the matrix P is updated to reflect the degree of estimation error present.   As the estimation process is repeated, the values of the P matrix decrease which

37

results in smaller Kalman gains being applied to the corrector equation.

e. **Procedure NEWEST**

NEWEST essentially implements the corrector equation resulting in a new predicted parameter vector $\underline{\theta}$. The prediction error, or innovation, is first calculated and the Kalman gains are applied to this quantity. The past prediction of the parameter vector is then updated by the calculated correction value. The K matrix, provided by the procedure KGAIN, minimizes the error between the predicted output and actual output. After repeated application of the corrector equation, the prediction error approaches 0 and the parameter vector $\underline{\theta}$ approaches the correct steady state values.

f. **Procedure INNERPROD**

This procedure produces the current output estimate through the product of the regression vector ϕ and the predicted parameter vector $\underline{\theta}$.

2. **Procedure GENERATE DIVEPLANE COMMAND**

The procedure GENERATE DIVEPLANE COMMAND calculates the diveplane command using variable structure controller technique. The required code is short and simple, requiring little explanation. This procedure calculates the scalar value σ through the vector product of the left eigenvector of the system and the estimated state vector. The diveplane command for the augmented system, δ_o, is then generated by applying a scaling factor to satsign(σ). The satsign function

38

is implemented using the subordinate program function "SAT".
Applying the relationship, $\delta = \delta_o - 0.32\theta$, results in the
diveplane command for the original system. At this point, a
software limit of 0.4 radians is placed on the diveplane
command in order not to exceed the physical travel limit of
the diveplane actuators. The limited value of δ is then
converted from radians to a voltage equivalent. The
conversion from the decimal voltage value to the binary
equivalent is performed by another subordinate program
function, "CONVERT ANALOG 2 DIGITAL". Finally, the binary
equivalent of the diveplane command is stored in the D/A
channel 0 input register before being applied to the vehicle
diveplane.

D.   DESCRIPTION OF PROGRAM FLOW

The modular design approach of using separate procedures
and functions to implement the digital autopilot program
significantly simplifies the design process. However,
evaluating procedural dependencies and program progression are
made more difficult by this approach. Figure 9 illustrates
the conceptual flow of the digital autopilot program.
Although not all procedures are included, those which are
crucial to understanding the operation of the control system
are labeled in Figure 9, and their functions described below.

- GET TARGET DEPTH - Upon entering the program control
  loop, the user is prompted to enter the desired target
  depth of the vehicle in feet. The desired depth is then
  used to calculate depth error.
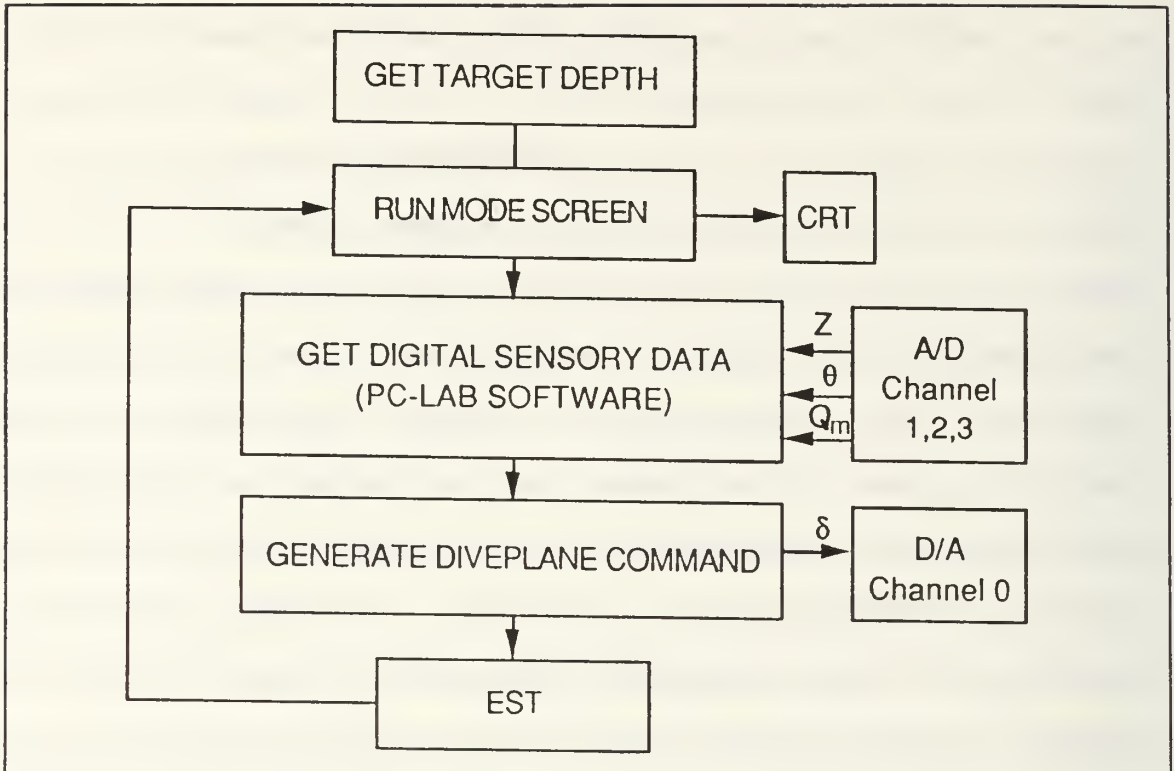
Figure 9   Conceptual Flow of Digital Control Program

- RUNMODE SCREEN – Writes vehicle parameters and lists user
  options on the screen. Information is updated at
  periodic intervals selected by the programmer.

- GET DIGITAL SENSORY DATA – Reads digitized sensory data
  from A/D data registers. Data is converted to physical
  units and stored for processing by the control program.
  The process is coordinated by PC-LAB software.

- GENERATE DIVEPLANE COMMAND – Computes diveplane command,
  digitizes this value, and stores result in D/A input
  register.

- EST – Provides prediction of system states.

VII. EXPERIMENTAL RESULTS

A. GENERAL

This work has been concerned with the theory and application of various control concepts which have resulted in the development of an integrated state estimation and control methodology for the AUV. Throughout the development of the estimator and controller, a number of variables which have an effect on controller performance have been left undetermined. This chapter details the process which lead to the selection of these variables as well as the final result of depth controller implementation.

B. RLS ALGORITHM RESULTS

The RLS algorithm was designed to provide crucial information pertaining to the pitchrate of the vehicle. With no pitch measurement available, the determination of vehicle pitch was also dependent on the RLS algorithm. To evaluate the performance of the RLS algorithm, the analog simulator was programmed with the continuous time coefficients which express the lag relationship between diveplane command and pitchrate. The values of the zero and pole were 2.53 and 1.8 respectively (representing vehicle dynamics at 2.1 ft/sec). The discrete estimates of the lag coefficients generated by the RLS algorithm are the first two elements of the parameter vector $\underline{\theta}$. By comparing these estimates to the corresponding discrete

41

equivalents of the lag filter coefficients, the performance of the RLS algorithm was examined. Figure 10 represents the plot of the estimated discrete equivalent of the pole of the model. Note that the value converges to 0.9 in approximately four seconds. The correct discrete value of the pole is 0.9143. The RLS algorithm converges to an extremely accurate estimate of the system pole in a reasonable amount of time. The estimated discrete equivalent of the zero, as shown in Figure 11, converges to a value of 0.139 in approximately seven seconds. The correct discrete value of the zero is 0.127. The estimate of the zero is not as accurate as that of the pole, which is generally the case for the RLS algorithm.

Although the approximation of the discrete system parameters may prove to be important in the determining the dynamic model of future vehicle designs, the most important objective in this work is the determination of the bias component of the pitchrate signal. The bias was simulated as a DC voltage added to the pitchrate voltage output of the analog simulator. Figure 12 represents the bias estimate for the system with an injected bias of 0.2 volts. The RLS algorithm produced a bias estimate which converged to the correct value of 0.2 in approximately eight seconds. Different values of pitchrate bias, ranging from one to three volts, were injected into the simulation and the corresponding bias estimate was observed. Varying the magnitude of the

42
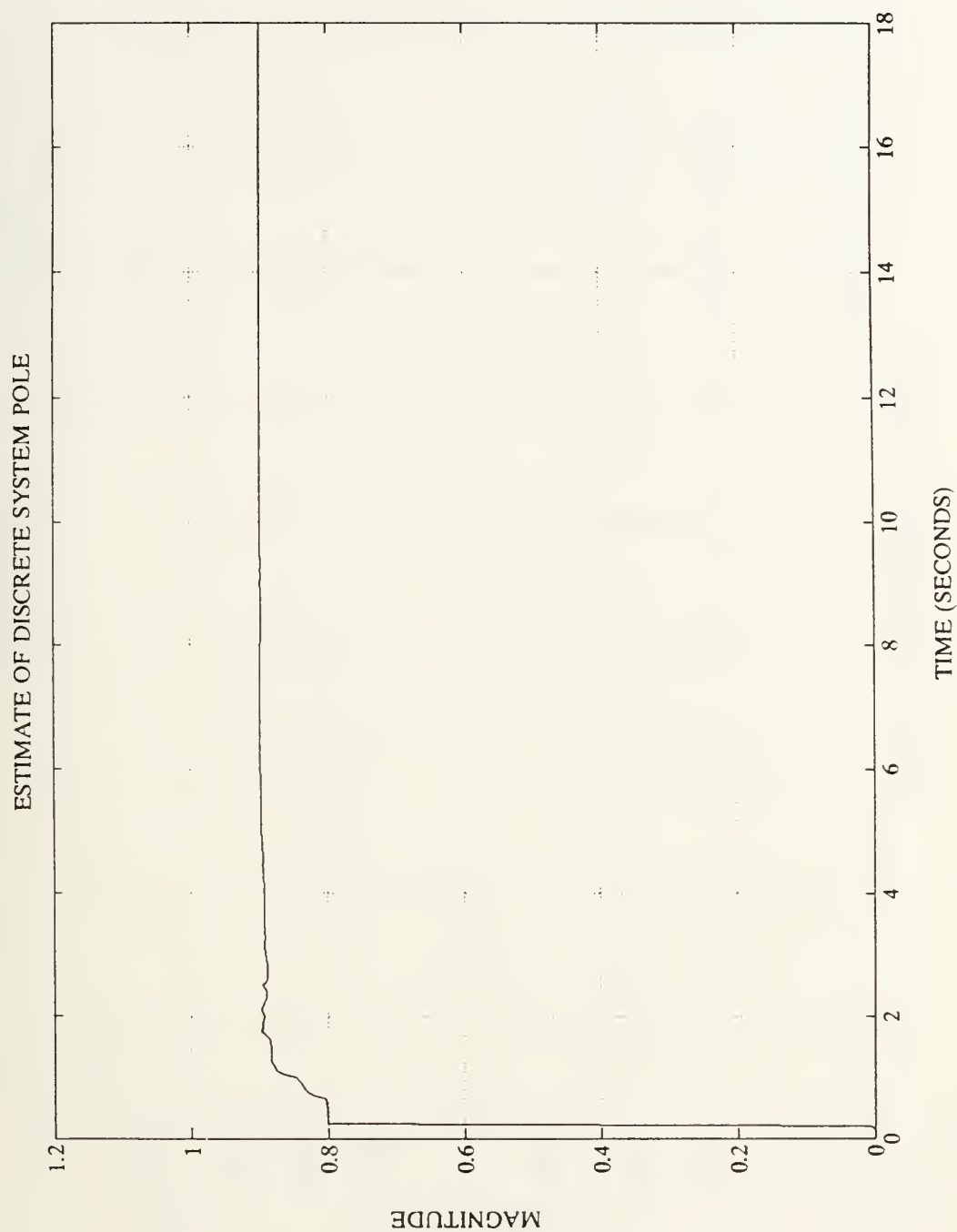
Figure 10   RLS Estimate of Pole

43
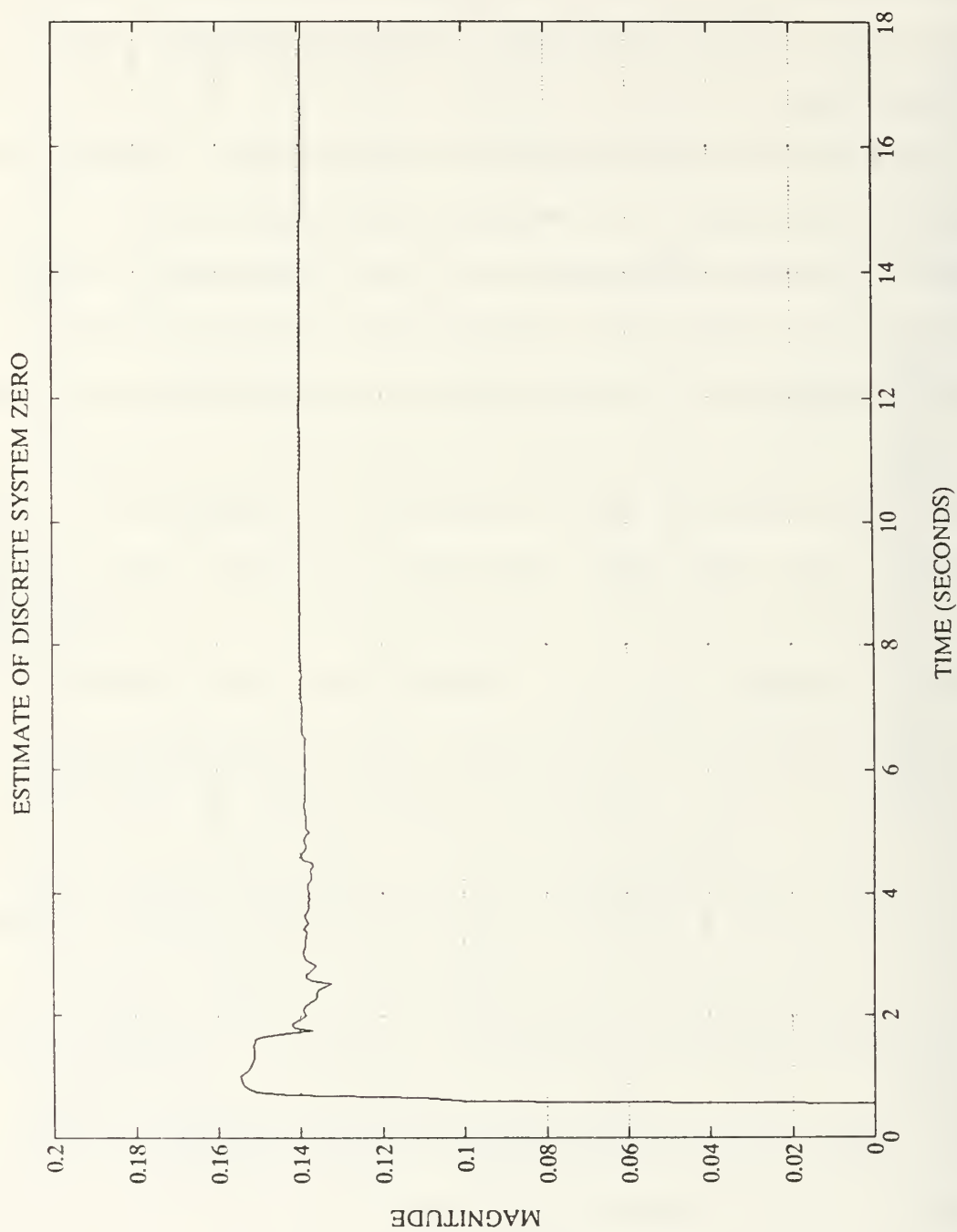
Figure 11   RLS Estimate of Zero

44
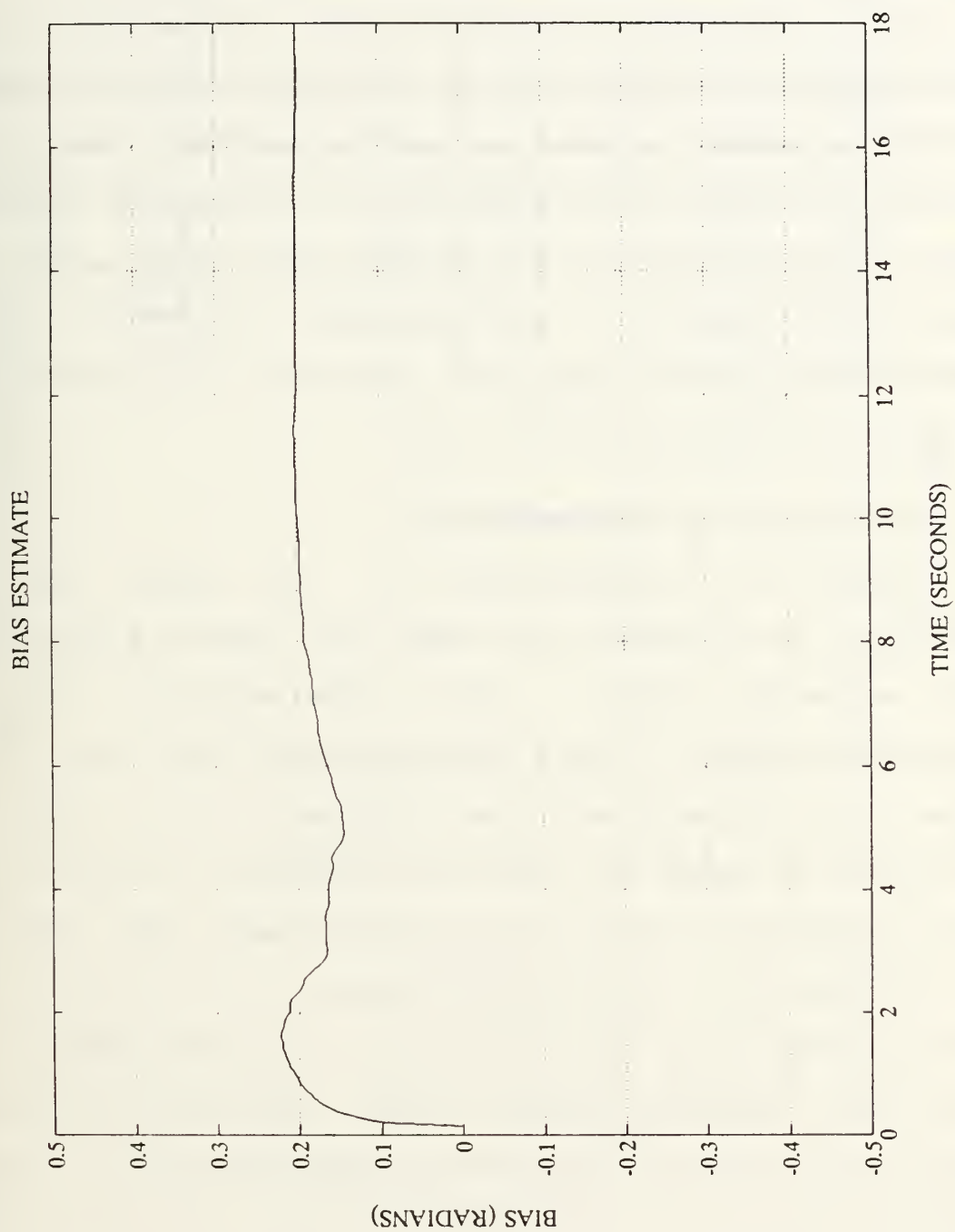
Figure 12   RLS Estimate of Bias

pitchrate bias produced no significant variations in the accuracy or convergence time of the algorithm.

After estimating the pitchrate bias, it was removed from the measured pitchrate signal in order to yield a pure signal for state feedback as well as input to the LQE. Figure 13 is a plot of the measured pitchrate and the estimated pitchrate with the bias removed. As is evident from the plot, the bias used in this case was 0.2 radians/second. The results clearly demonstrate that this approach effectively removes the bias from the measured pitchrate signal.

C. RESULTS OF LQE IMPLEMENTATION

Design and implementation of the Linear Quadratic Estimator was intended to be the least difficult portion of the estimator design. Upon experimentation with the controller design, it was discovered that this was not the case. As previously mentioned, the feedback gains applied to the pitch and depth estimates were determined by the ratio of the covariance of system noise to measurement noise specified by the designer. In this application, an increase in this ratio suggests less confidence in the pitchrate input to the LQE rather than the measured output (depth) of the vehicle. The specified noise covariances do not represent the actual noise present in the corresponding signals, but only providea means of determining the optimal steady-state Kalman gains to be applied to the observer. The general approach was to
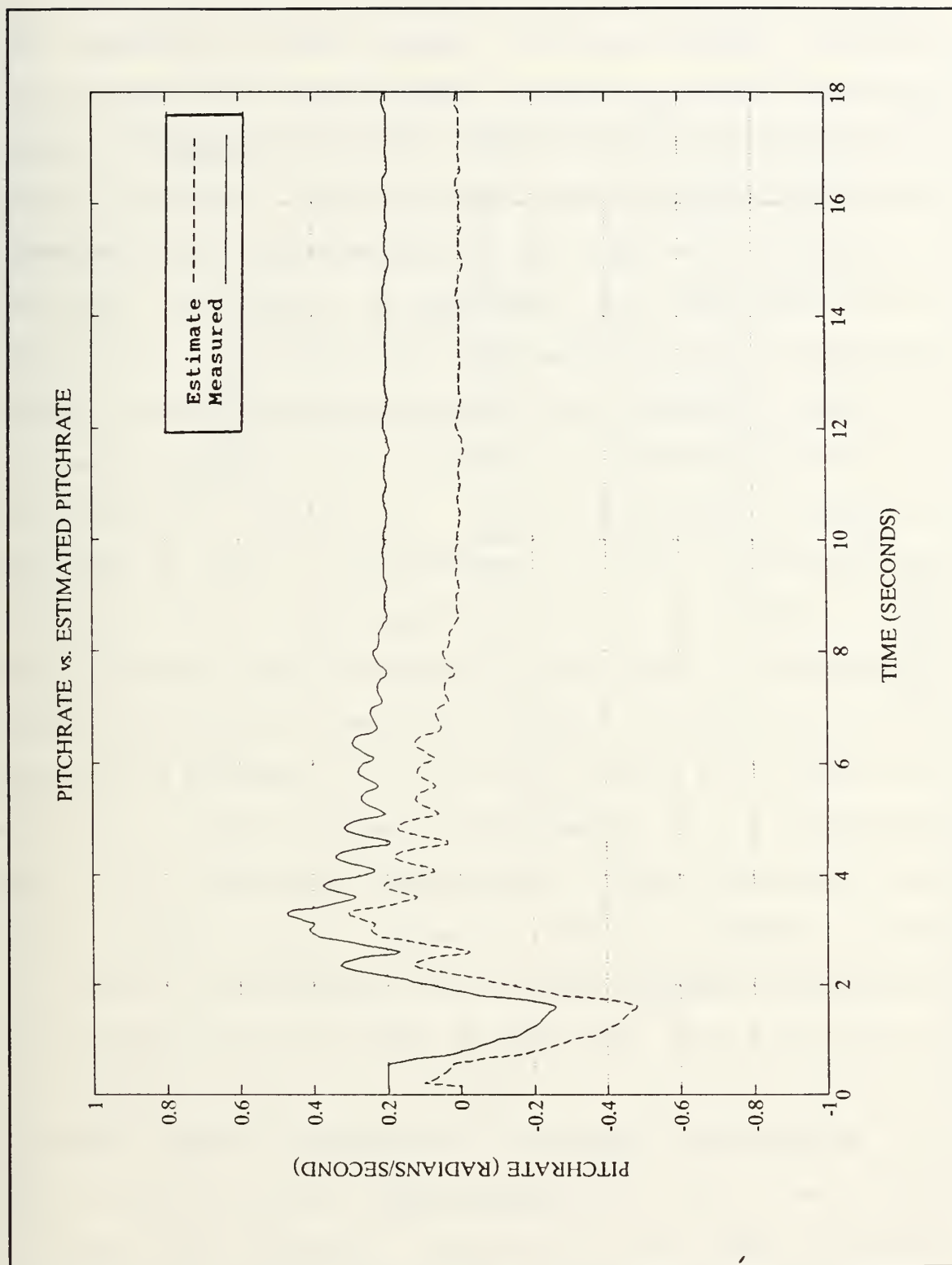
Figure 13   Pitchrate with Bias vs. Pitchrate with Bias Removed

47

generate several sets of Kalman gains associated with different noise covariance combinations and implement the gains which provided the best result. The results achieved through this appoach were less than ideal. Selecting a small ratio between the input noise covariance (S_t) and measurement noise covariances (R_t) resulted in an observer which was incapable of tracking the pitch of the vehicle. As the noise ratio was increased, the tracking performance of the observer increased substantially; however, the observer became too sensitive to estimation error. The effect of varying the ratio between S_t and R_t is demonstrated in Figure 14 and Figure 15. In Figure 14, S_t and R_t were specified to be 6.0 and 1.0, respectively. Note that the response of the observer is too slow to provide for accurate tracking of the vehicle pitch. In Figure 15, S_t is 100.0 and R_t is 1.0, resulting in increased magnitudes of the corresponding feedback gains. As can be seen from this result, the increased magnitude of the Kalman gains yielded a faster observer which tracked pitch considerably better than the case represented in Figure 14. On the other hand, the negative effect of the increased gains on the performance of the observer were equally obvious. As the gains were increased, the observer became extremely sensitive to any error between the estimated depth and the measured depth of the vehicle. Although this error is minimal, it is amplified by the feedback gain applied to the observer resulting in a noisy pitch estimate. However, it was
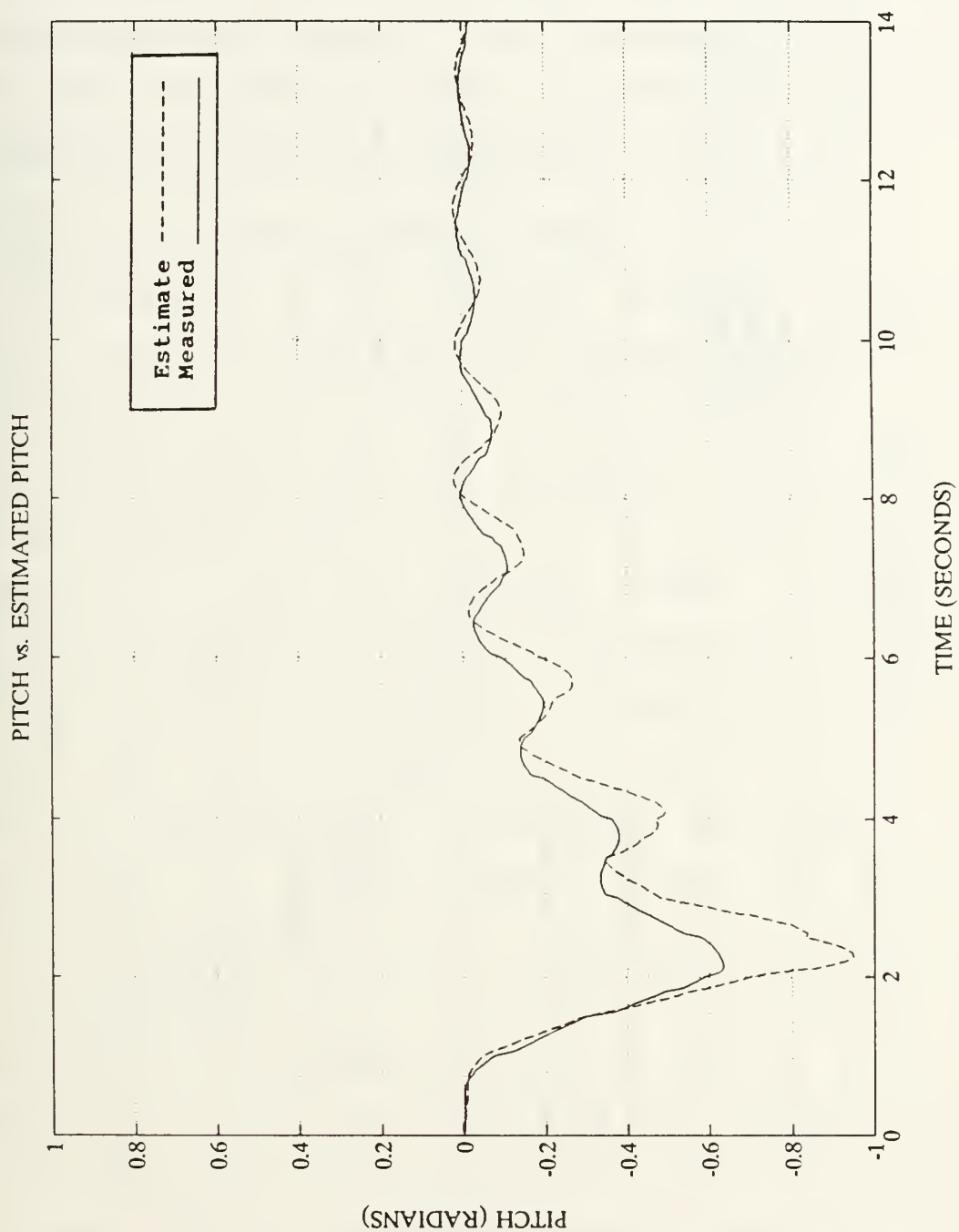
Figure 14   Pitch Estimate with $S_t = 6.0$ and $R_t = 1.0$
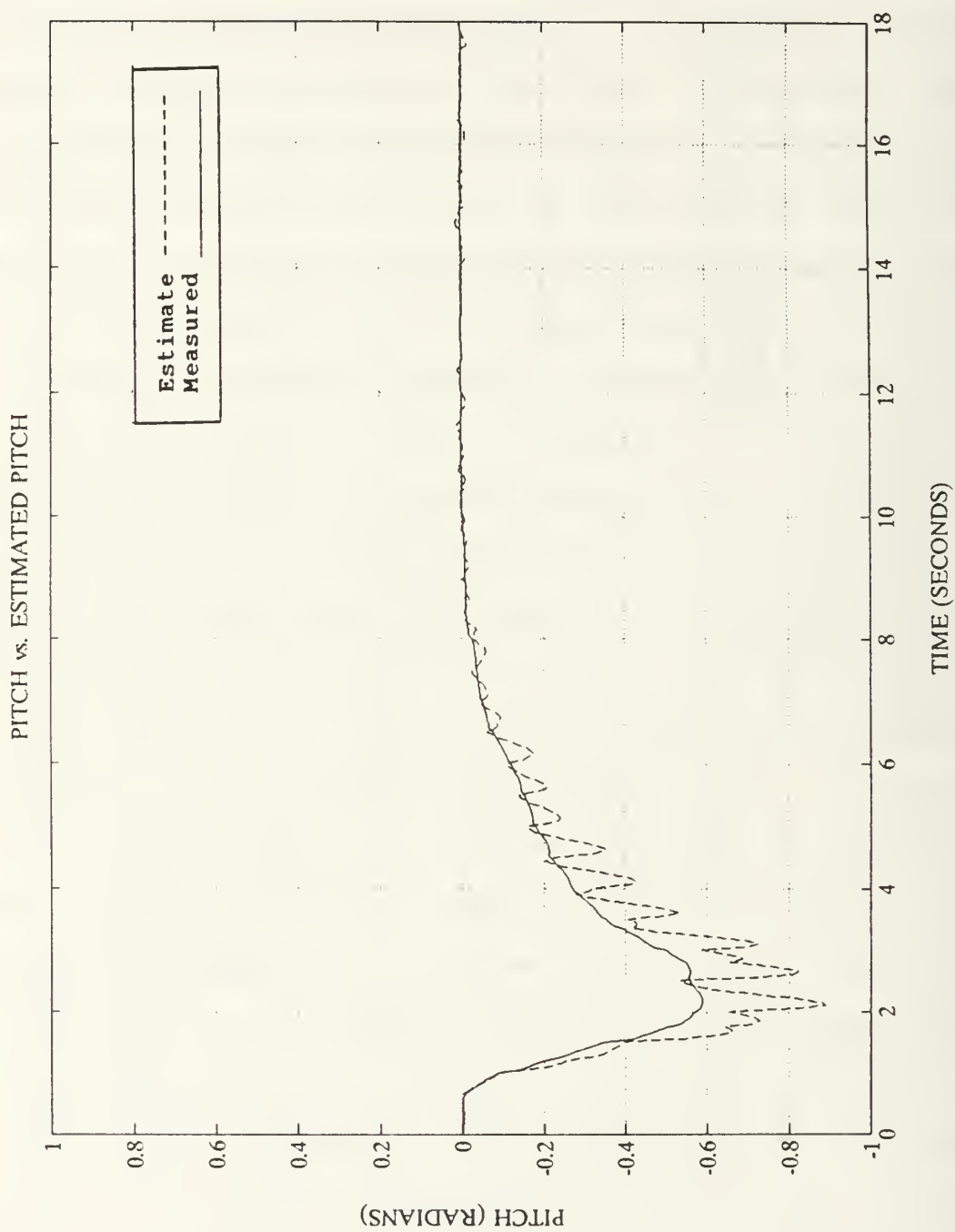
Figure 15   Pitch Estimate with S_t 100.0 and $R_t = 1.0$

50

determined through further experimentation that the sensitivity of the observer had little affect on the depth trajectory of the vehicle. As a consequence, the case represented in Figure 15 was selected in the final implementation of the state estimator.

D. VARIABLE STRUCTURE CONTROLLER RESULTS

Implementation of the variable structure controller involved extensive experimentation in order to determine the best combination of selectable controller parameters. The parameters in question are the saturation value, K, of the satsignum function and the width of the linear region of operation, $\Delta\sigma$. Since the diveplane command is ultimately limited to 0.4 radians by physical considerations, varying K has little effect on the diveplane command applied to the vehicle. The value of K was set to a value of two for the controller implementation. In contrast, the width of the linear region of operation greatly affects the response characteristics of the controller. Selection of a small value of $\Delta\sigma$ results in a more rigid controller characterized by longer periods of diveplane saturation as well as diveplane oscillations as the vehicle maintains the commanded depth. The effect on the vehicle depth trajectory was an overshoot of the desired depth followed by repeated corrections. Once attained, the commanded depth was maintained at the expense of continuous diveplane commands. In contrast, large values

51

of $\Delta\sigma$ resulted in a sluggish vehicle response. The diveplanes rarely entered the saturation region and therefore, the vehicle was slow to reach the desired depth. Once at the ordered operating depth, the vehicle was unable to maintain depth within a reasonable degree of accuracy. The optimal value of $\Delta\sigma$ was obtained through experimentation with different values of $\Delta\sigma$ and evaluation of the corresponding results. The underlying objective of the controller implementation was to provide a robust trajectory to the desired depth and maintain that depth with minimum diveplane command applied. Figure 16 depicts the plot of the diveplane command generated for a dive to five feet using 1.0 as the selected value of $\Delta\sigma$. The corresponding depth trajectory is contained in Figure 17. Note that the diveplane is initially saturated directing the vehicle toward the desired depth. As the vehicle approaches the desired depth, the diveplane command provides corrections to the vehicle trajectory. Once achieved, the ordered depth was maintained with minimal control input. The controller provided the best response with $\Delta\sigma$ set at 1.0 and, as a result, this configuration was used in the final controller implementation.
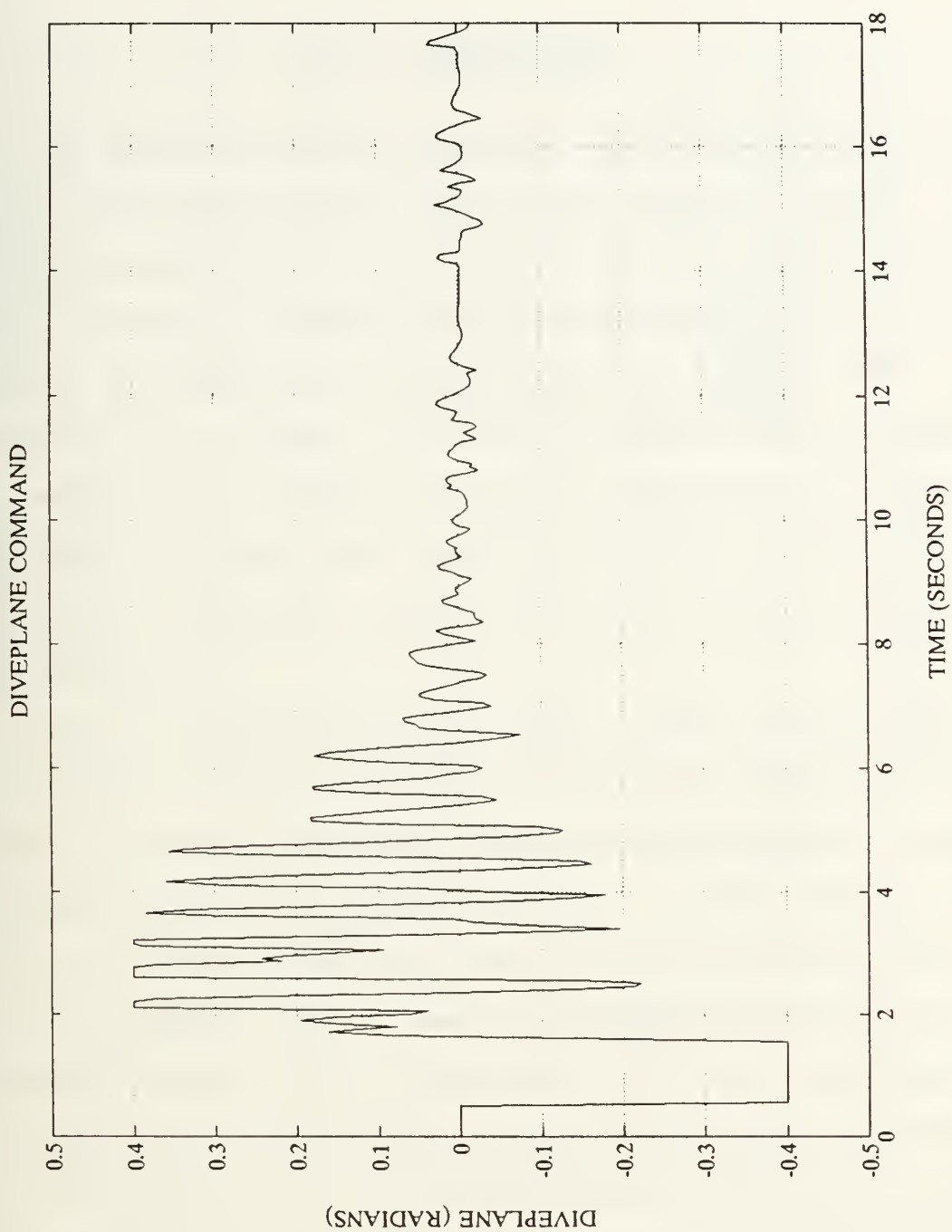
Figure 16   Diveplane Command for a Dive to 5 Feet
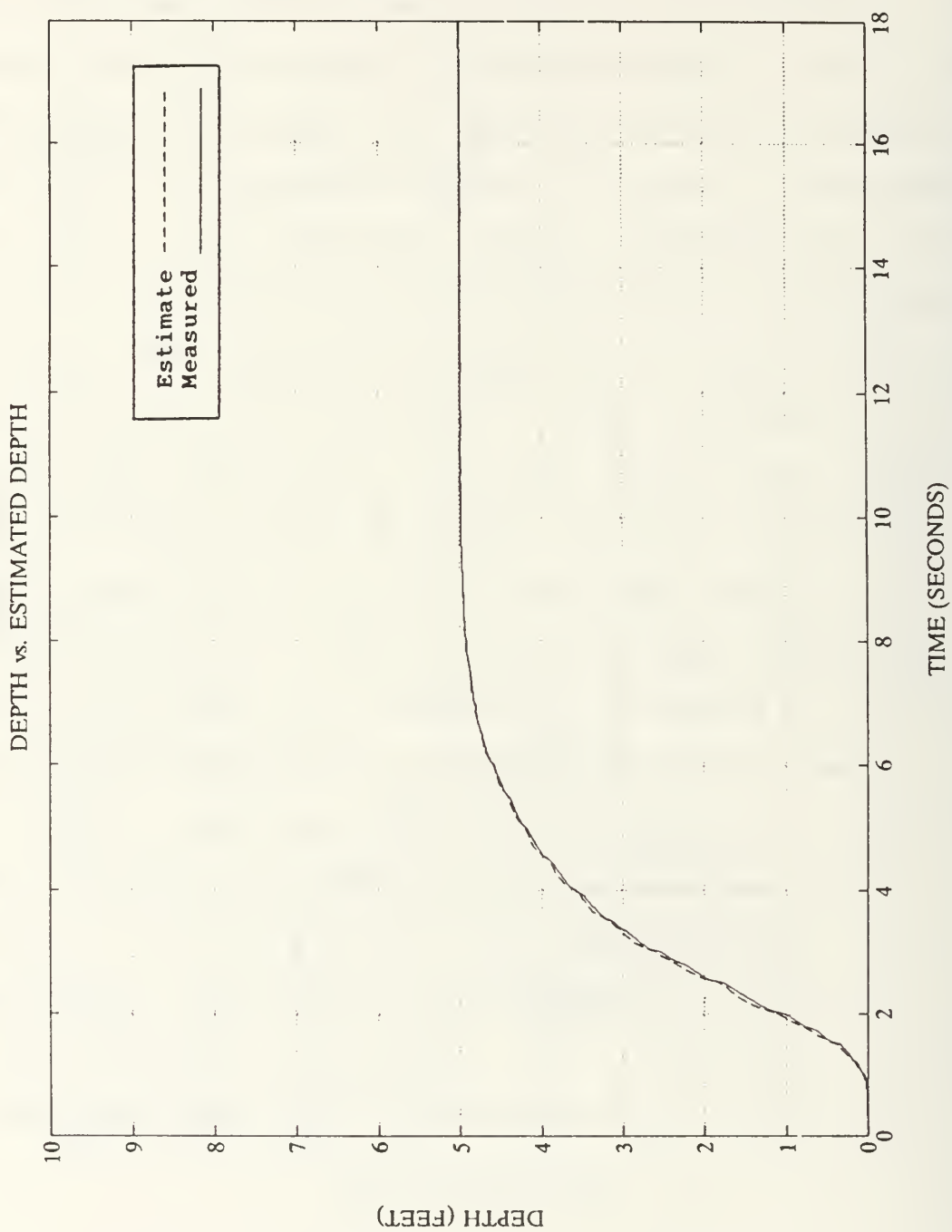
Figure 17   Depth Trajectory for a Dive to 5 Feet

VIII. <u>CONCLUSIONS</u>

The objective of this work has been to design a digital controller which provides robust depth control of an AUV. The success or failure of this research is based on the ability of the design to provide accurate estimates of the system states, and generate a closed-loop control input which is a function of the states. Although the digital control program was tested using an analog simulation rather than the intended prototype vehicle, the results obtained provide for a reasonably accurate analysis of the design concepts implemented.

Having only specified the order of the linear model of the system, the RLS algorithm estimated the model parameters within a reasonable degree of accuracy and provided an equally accurate estimate of pitchrate bias. Unfortunately, the varying dynamics of the AUV could not be simulated using the analog computer. As a result, the performance of the RLS algorithm could not be evaluated under the conditions of changing speed and acceleration which the vehicle is subjected to during maneuvering. From the results obtained, however, it would be reasonable to conclude that the algorithm would perform satisfactorily under these varying conditions.

The design of the pitch estimator was intended to be simple while still providing a reasonable estimate of vehicle pitch. As discussed in Chapter VII, the LQE did not perform as originally intended. The estimator proved to be very sensitive to measurement noise, resulting in a noisy pitch estimate. The negative effect of the noisy pitch estimate on system performance was realized as high frequency oscillations in the diveplane command. Fortunately, the effect on the resulting depth trajectory of the vehicle was minimal.

In the final analysis, the combination of an adaptive parameter estimation technique and variable structure control provided an effective means of vehicle depth control. Even in the presence of considerable state estimation error produced by the LQE, the variable structure controller directed the vehicle along a stable trajectory to the desired depth. Due to the flexibility of the controller design, virtually any vehicle response characteristic can be obtained by altering the controller parameters. Considering the wide range of possible missions of the AUV, this particular attribute of the variable structure controller might prove to be useful.

DIGITAL CONTROL PROGRAM

```
{$c-}
program    AuvAutoPilot  ( input, output );

{ TITLE       : AUV Automatic Control Program (AUTOCON3.PAS)
  AUTHOR      : J.M. WILLIAMS
  APPLICATION : Test of real-time controller for the AUV
  DATE        : 29 Aug 1989

  Project Description : This program implements digital
control of the NPS autonomous underwater vehicle (AUV) in
the vertical or dive plane.  It samples vehicle sensor input
from three channels : depth, pitch, pitchrate.  The depth
signal is then passed to a DepthError module which compares
the actual sensor depth with a model reference depth
simulated by a depthgain.  An depth error voltage is then
generated and passed to a GenerateDivePlaneCommand module
which processes the error signal and sends out an
appropriate command to the diveplane actuators. The gains in
the algorithms reflect the discrete transfer function gains
for diveplane command response derived from vehicle
identification analysis.      }


{ -------------GLOBAL DECLARATIONS--------------- }

const

{ ----------- Screen declarations -------------- }

    x1              = 5;       { Upper left corner : left edge }
    y1              = 2;       { Upper left corner : upper edge}
    x2              = 75;    { Lower right corner : right edge}
    y2              = 24;    { Lower right corner : bottom edge}


type
  str10 = string [10];
  str60 = string [60];

var
  hr,hr2,min,min2,
  sec,sec2,hun,hun2              : byte;
  seconds                       : real;
```

{----------- INCLUDED FILES Declarations ---------------- }

{$I pcldefs.tp }   { PC LAB Trubo Pascal routines.}

{$I pclerrs.pas }   { PC LAB error code messages file.}

{$Iinidac.auv   }
{ This procedure initializes the DT 2801-A TO ZERO VOLTS AND
MUST BE EXECUTED BEFORE THE AUV IS HOOKED TO THE COMPUTER.}

{$Igettime.auv   }
{ No arguments; returns hr, min, sec, hun : byte}

{$Ishotmdff.auv }
{ Input the output of TimeDiff.auv and this procedure
displays the time difference between the two most current
Get Time.auv results.
  ShowTimeDifferene (x:integer) . }

{$Itimediff.auv}
{ Input: hr,hr2,min,min2,sec,sec2,hun,hun2 from two calls
  of GetTime.auv and this returns the difference in
  seconds as a REAL variable.}

{$Idrawbox2.auv}
{ Input x1,y1,x2,y2 : integer to specify the corner limits
of the box. This procedure clears screen and draws a
rectangular box of specified dimension using ASCII double
line characters.
          .
{$Iclrbox2.auv}
{ Input x1,y1,x2,y2 : integer to specify the corner limits
of the box. This procedure uses a FAST means of clearing a
box of specified dimension. The box dimension should be
delcared as constants.}

{$Iboxprint.auv }
{ Input the printrow, leftboxedge, rightboxedge : integer
and printstring :str60. This procedure centerprints the
string in the box at the printrow specified without
overwriting the box border.}

{$Ishowfast.auv}
{ Input message : str60, column,row : integer. To specify
the x,y position on the screen for a FAST message print.}

{$Ikeyhit.auv}
{ This is a boolean function which returns true or false if
key is pressed; it also returns keycode replies VAR reply,

58

```
{$Itabxy.auv}
{ Input tabcol,tabrow : integer; like gotoxy}

{Iboxpause.auv}
{ Input xpause,ypause : integer to specify where "Press any
key to continue" message is to be printed.}

{$Igetkey.auv }
{ Input as a string of chars, the set of acceptable replies;
ie 'YyNn'.  This procedure waits until one of the acceptable
replies has been entered.}

{$Iutils.auv   }
{ Included are some housekeeping and debugging routines.}

{$Iconvadv.auv }
{ Includes functions to convert depth,speed and pitchrate to
vehicle values. }

{ ************* MAIN PROGRAMS PROCEDURES   *************** }


{ ************* USER INTERFACE MODULES   *************** }

procedure MainMenu ( var reply : char );

{  This procedure presents the AUV screen and solicits an
option to Run the AUV from the StatusAndCommand procedure or
to Quit.}

begin
   repeat
     clrscr;
     drawbox2(x1,y1,x2,y2);
     boxprint(y1+3,x1,x2,'N A V A L    P O S T G R A D U A T
             E   S C H O O L');
     boxprint(y1+5,x1,x2,'D E P A R T M E N T    O F ');
     boxprint(y1+6,x1,x2,'M E C H A N I C A L    E N G I N E
             E R I N G');
     boxprint(y1+8,x1,x2,'AUTONOMOUS    UNDERWATER
             VEHICLE');
     boxprint(y1+10,x1,x2,'DIGITAL   AUTOPILOT   CONTROL
             PROGRAM');
     boxprint(y1+12,x1,x2,'**************************');
     boxprint(y1+15,x1,x2,'Do You want to RUN this program
             ..');
     boxprint(y1+16,x1,x2,'or Do You want to QUIT and return
             to DOS ?');
     boxprint(y1+20,x1,x2,'>>>>  ENTER Q OR R  <<<<');
     getkey ('QqRr',reply,reply2);
   until ( reply in ['Q','q','R','r'] ) and (reply2 =
   chr(0));
```

```
end;

procedure StatusAndCommand ( var mode : char );
{ This procedure begins the control program  screen.}

var
    mode2                    : char;

procedure StatusAndCommandScreen;

{  This is the status and control screen and solicits a user
input of F1 to RUN the program or Q to Quit and exit to the
main menu.}

        { --------- StatusAndCommandScreen ---------- }
begin

      clrbox2 (x1,y1,x2,y2);
      boxprint(y1+1,x1,x2,'AUV STATUS / COMMAND AND CONTROL
            SCREEN');
      boxprint(y1+2,x1,x2,'=============================');
      boxprint(y1+7,x1,x2,'CHOOSE YOUR DESIRED CONTROL MODE
            :');
      boxprint(y1+9,x1,x2,'ENTER KEY  << F1 >>    TO START
            AUV CONTROL');
      boxprint(y1+11,x1,x2,'ENTER  << Q >> TO QUIT AND
            RETURN TO MAIN MENU');
      boxprint(y1+16,x1,x2,'PRESS EITHER   F1   OR   Q');

end;           { -------- StatusAndCommandScreen --------}

{ ********** CLOSED LOOP CONTROL ROUTINES *************** }

Procedure ClosedLoopControl;
{ This module comprises the closed loop control scheme.}

label 5;
label 1;
const
     maxdepth        = 33;
     mindepth        = 0;
     updateincrement = 10;

type
    try                 = array[1..10] of real;
    activecontrolmode   = ( run, reset, exit );
    allowabledepthrange = mindepth..maxdepth ;
    auvattitude         = ( climb, maintain, diving );
    digitalintegerarray = array [1..3] of integer;

var
filename:string[14];
```

60

```
filevar:text;

auvdepth,auvdepthvolts,auvspeed,auvpitch,yawrate,diveplane,
auvspeedvolts, auvpitchrate,auvpitchratevolts,estdepth,err,
ac1,ac2,ac3,ac4,ac5,ac6,deptherrorvolts, targetdepthvolts,
bias1,divevolts,tgtnew,targetdepth,speed:real;

    adv                              :digitalintegerarray;
    j,status,time                    : integer;
    modereply,modereply2             : char;
    activemode                       : activecontrolmode;
    updatecounter,initial            : integer;
    depthrange                       : allowabledepthrange;
    attitude                         : auvattitude;


procedure GetTargetDepth (var tgtdepth : real ;
                          var tgtdepthvolts : real );

{ This procedure solicits the target AUV operating depth and
converts it to an AUV equivalent targetdepth analog voltage
and passes both of these parameters. }


begin     { --------  GetTargetDepth -------- }

    clrbox2 (x1,y1,x2,y2);
    boxprint(y1+10,x1,x2,'ENTER THE A U V TARGET OPERATING
             DEPTH');
    boxprint(y1+11,x1,x2,'NOTE : THE DEPTH SHOULD BE IN FEET
             (0.0 - 10.0)');
    repeat
       begin
          boxprint (y1+13,x1,x2,'ENTER THE TARGET OPERATING
                    DEPTH ');
          gotoxy (x1+33,y1+15);
          read ( tgtdepth );
       end;
    until   tgtdepth >= 0.0 ;

    tgtdepthvolts := tgtdepth ;

end;        { --------  GetTargetDepth ----------- }
 procedure RunModeScreen;

{ This procedure displays the Closed Loop Control Screen in
   the RUN MODE.    }

begin          { --------------- RunModeScreen ----------- }

     clrbox2 (x1,y1,x2,y2);
     boxprint(y1+1,x1,x2,'A U V   S T A T U S   /   C O N T
```

```
                    R O L   S C R E E N');
     boxprint(y1+2,x1,x2,'==============================');
     boxprint(y1+4,x1,x2,'STATUS OF A U V OPERATING
             PARAMETERS :');
     write (tabxy (x1+5,y1+6),'AUV DEPTH      [feet] : ');
     write (tabxy (x1+5,y1+7),'AUV PITCH      [rads]  : ');
     write (tabxy (x1+5,y1+8),'AUV PITCHRATE [rads/sec]:
         ');
     write (tabxy (x1+5,y1+9),'AUVDIVEPLANE   [rads]  : ');
     write (tabxy (x1+5,y1+10),'BIAS =          :');
     boxprint(y1+11,x1,x2,'A U V  CONTROL STATUS :');
     write (tabxy (x1+5,y1+13),'CURRENT TARGET DEPTH : ');
     write (tabxy (x1+5,y1+14),'CURRENT MODE         : ');
     write (tabxy (x1+5,y1+15),'CURRENT MANEUVER     : ');
     boxprint(y1+18,x1,x2,'PRESS KEY   F1 .. TO ENTER NEW
             TARGET DEPTH.        ');
     boxprint(y1+19,x1,x2,'PRESS KEY   F3 .. TO EXIT ACTIVE
             CONTROL.             ');

     end;       { ------------RunModeScreen---------- }

procedure
UpdateRunModeScreen(updatedepth,updatepitch,updatepitchrate,
updatediveplane                         : real;
                    updatetargetdepth : real;
                           updatemode: activecontrolmode;
                    updateattitude : auvattitude;
                          var bias1: real);

{  This module updates the Closed Loop Control Run Mode
Screen with updated display parameters. Updates occur in
intervals specified by updateincrement interval declared in
ClosedLoopControl procedure.                            }

begin       { --------  UpdateRunModeScreen ---------  }

 { UPDATES    STATUS OF A U V OPERATING PARAMETERS }
   writeln (tabxy (x1+37,y1+6),updatedepth:6:3);
   writeln (tabxy (x1+37,y1+7),updatepitch:6:3);
   writeln (tabxy (x1+37,y1+8),updatepitchrate:6:3);
   writeln (tabxy (x1+37,y1+9),updatediveplane:6:3);
   writeln (tabxy (x1+37,y1+10),bias1:8:6);

 {      UPDATES THE   A U V  CONTROL STATUS           }
   write (tabxy (x1+30,y1+13),updatetargetdepth:6:2);
   case updatemode of
      run : writeln (tabxy (x1+30,y1+14),'RUN  ');
      exit : writeln (tabxy (x1+30,y1+14),'EXIT ');
   end;
   case updateattitude of
      maintain : writeln (tabxy (x1+30,y1+15),'MAINTAINING
                          DEPTH          ');
```

62

```
          climb     : writeln (tabxy (x1+30,y1+15),'CLIMBING TO
                                TARGET DEPTH');
          diving    : writeln (tabxy (x1+30,y1+15),'DIVING TO
                                TARGET DEPTH  ');
      end;

end;        {-------------UpdateRunModeScreen---------------}

procedure GetDigitalSensoryData(var
depthanalogvolts,pitchanalogvolts,
                    pitchrateanalogvolts :real) ;

{  This procedure uses PCLAB routines to sample selected
input telemetry channels from the AUV and digitizes these
inputs and multiplies them by the specified gains.

    DT 2801-A / DT 707 Board set up:

                            channel 1  - AUV depth input
                            channel 2  - AUV pitch input
                            channel 3  - AUV pitchrate input}


const

{ These are artificial gains used to simulate AUV telemetry
during program development.  One signal from a signal
generator (+/- 1.25, 9 Hz, characteristic of the pitchrate
signal) is input to all 3 input channels.  Gains are applied
to simulate the actual values.  These and their application
in the procedure body should be removed after program
development is completed.}

      depthgain     = 1.0;
      pitchgain     = 1.0;
      pitchrategain = 1.0;

{ These are AUV to DT 2801-A / DT 707 hook up board channel
configurations, conversion and computational arguments.}

depthchannel  = 1;        { AUV output to DT-707 input channel
                            assignment   }
depthpfs      = +10.0;  { Peak depth signal value}
depthmfs      = -10.0;  { Minimum depth signal value}

pitchchannel  = 2;        { AUV output to DT-707 input channel
                            assignment   }
spdpfs        = +10.0;  { Peak speed signal value}
spdmfs        = -10.0;  { Minimum speed signal value}
pitchratechannel = 3;   { AUV output to DT-707 input channel
                            assignment   }
pitchratepfs  = +10.0;  { Peak pitchrate signal value}
```

63

```
pitchratemfs   = -10.0;   { Minimum pitchrate signal value}

noc            = 4096;    { Number of Codes; conversion
                            resolution. The DT 2801-A performs
                            a 12 bit conversion. NOC = (2 ^ 12
                            conversion bits), ie 4096}

{ SetUpAdc and ADConTrigger PCL function arguments
                                : p 6-8 PCL  documentation  }


    boardnum        = 1;
    numa2dchan      = 3;
    timingsource    = 2;    { -- Sets trigger,internal clock }
    adcgain1        = 1;
    adcgain2        = 2;
    adcgain4        = 4;    { Sets the A/D gain; 1}
    adcgain8        = 8;
    startchannel    = 1;
    endchannel      = 3;

var
    pitchadv,
    depthadv,
    pitchrateadv,
    signaladv,                      { Signal analog data value}
    counter,status,
    chanum,i,j                      : integer;


begin     { -------- procedure GetDigitalTelemetry  --------}

            { Set up the DT 2801-A board to take data.}

    status := SelectBoard (boardnum);

{ Set up the DT 2801-A board to take data from 3 input
channels; Data sampling is initiated by the ADConTrigger
single channel sample of the depth channel and then single
ADC value samples of the speed and pitchrate follow.   The
Trigger is connected to the DT 707 board at terminal 49 from
a signal generating source.}

status := ADConTrigger ( depthchannel, adcgain1, depthadv );
status := ADCValue ( pitchchannel, adcgain1, pitchadv );
status := ADCValue ( pitchratechannel,adcgain1,
                    pitchrateadv);


{ Convert the digitized Analog Data Values for speed, depth,
pitchrate to analog voltage values.  The algorithm for this
conversion is found in Appendix D of the PCLAB
```

64

documentation. }

```
   depthanalogvolts := ( depthadv * (depthpfs-depthmfs)/noc )
                       + depthmfs;
   {this allows for a -1 volt bias in the depth cell reading
    at zero depth}
   pitchanalogvolts := ( pitchadv * (spdpfs-spdmfs)/noc )
                       + spdmfs;
   pitchrateanalogvolts := ( pitchrateadv * (pitchratepfs -
                          pitchratemfs)/noc )+pitchratemfs;


end;    { --------- procedure GetDigitalTelemetry  ------ }

procedure ATTITUDE_ ( tdepthvolts, adepthvolts : real;
                      var attitude : auvattitude    );
```

{ This module represents the "AUV Model Reference State
Space." Actual depth telemetry and the target depth are
compared and a voltage difference is computed. This
difference is then "dropped" through a voltage filter
to determine if the difference if within an acceptable
tolerance, or if a corrective diveplane command is
necessary. A "model gain" is applied to the voltage
difference and an errorvoltage is calculated and passed to
the main program for dive command generation.   Although
these parameters are single valued, in a multi-state control
program these parameters could be implemented as arrays and
the model gain array could be the result of a real-
time program running synchronously with the main control
program. }

{ COMPUTATIONAL SIGN CONVENTION: The voltage difference is
computed as the difference between TARGET DEPTH , or desired
AUV depth, and the ACTUAL DEPTH. PLUS voltage DIFFERENCE
generates down dive plane command; MINUS voltage DIFFERENCE
generates an UP dive plane command.}


```
const

   depthcontroltolerence =   0.1;
   modelgain             =   1.0;
var

    voltsdifference       : real;


begin         { ------ Errorvolts -------- }

   voltsdifference := tdepthvolts - adepthvolts;
```

```
{++++++++++ Control voltage filter   +++++++++++++++ }

{ These conditions check if depth is within tolerence.   If
so a zero error is assigned so as to result in a zero
diveplane command.}

  if  ( voltsdifference > 0) and
        ( abs(voltsdifference) <= depthcontroltolerence )
     then
       begin
          attitude := maintain;
       end
    else if  ( voltsdifference < 0) and
        ( abs(voltsdifference) <= depthcontroltolerence )
  then
       begin
          attitude := maintain;
       end

{ This condition checks if actual depth is less than target
+ tolerence.  In this case a DIVE  command is necessary to
correct depth.               }

    else if  ( voltsdifference > 0) and
            ( abs(voltsdifference) > depthcontroltolerence )
     then
       begin
          attitude := diving;
       end

{ This last condition checks to see if the actual depth is
more than target + tolerence.  In this case climb command is
necessary to correct depth.            }

    else if  ( voltsdifference < 0) and
            ( abs(voltsdifference) > depthcontroltolerence )
      then
        begin
          attitude := climb;
        end;

end;             { ------ Errorvolts ----- }
procedure EST(var up,yq,z:real;var vhat: real; var initial:
            integer; var bias,a1,a2,a3,a4,a5,a6: real);

      label  3;
 type
      try = array[1..10] of real;
      trytry = array[1..10,1..10] of real;
      try40 = array[1..40,1..40] of real;
      tryt40 = array[1..40] of real;
```

66

```pascal
        array20 = array[1..20] of real;
        matrix20 = array[1..20,1..20] of real;

const
        ns = 1;

var
        beta,khat,fc,xhat,g3,k,xu,xy,xz,xs,xnew,phi: try;
        spt,f3,sq : trytry;
        nx,np,ny,nf,nsnf,time,i,j : integer;
        uf,yf,y,yhat,ud,u,yd,p0,sr,sigmax,fs: real;

        procedure initializeArrays;

          var
              i,j: integer;
            begin
                  for i:= 1 to 10 do
                    begin
                       beta[i]:=0.0;
                       khat[i]:=0.0;
                       xhat[i] := 0.0;
                       xu[i]:=0.0;
                       xy[i]:=0.0;
                       xz[i]:=0.0;
                       xnew[i]:=0.0;
                       xs[i]:=0.0;
                     end;
                  np := 2*ns;
                  nx := np + 1;
                  ny := 1;
                  y:= 0.0;
                  uf:= 0.0;
                  ud:=0.0;
                  yd:=0.0;
                  yf := 0.0;
                  time := 0;
                  sigmax := 0.10;
                  sr := 1.0;
                  p0 := 1.0e6;
                  nf := 1;
                  bias := 0.0;
                  fc[1] := 0.8;
                  fc[2] := 0.2;
                  vhat  := 2.1;
                  fs    := 20.0;
                  for i:= 1 to nx do
                  begin
                        for j:= 1 to nx do
                        begin
                             phi[i] := 0.0;
                             sq[i,j] := 0.0;
```

67

```
                          spt[i,j] := 0.0;
                              if i = j then
                              begin
                              sq[i,j] := sigmax;
                              spt[i,j]:= p0;
                              end;
                   end;
              end;
         end;


    procedure innerprod(var y9: real;var phi9,x9:try;var
                  nx9:integer);

        var
             j: integer;
              begin
             y9:= 0.0;
                for j := 1 to nx9 DO
                begin
                     y9 := phi9[j] * x9[j]+ y9;
                end;
              end;




    procedure update(var phi8: try;var u8,y8: real;
             ns:integer);


        var
             n2:integer;
             i: integer;
             phisave: try;

        begin
              n2:=ns*2+1;
              for i:=1 to 2 do
             begin
              phisave[i]:=phi8[i];
             end;
                i := ns;
              while i >= 2 do
                     begin
                         phi8[i]:= phisave[i - 1];
                         phi8[ns + i] := phisave[ns+i-1];
                         i := i - 1;
                     end;
                     phi8[1]:= y8;
                     phi8[ns + 1] := u8;
              end;

                          68
```

```
procedure filter(var u4,y4:real;var x4,fc4:try;var
          nf4:integer);
      var
         nf42 : integer;
         savex4:try;

         begin
              update(x4,u4,y4,nf4);
              nf42:= 2 * nf4;
              innerprod(y4,x4,fc4,nf42);
         end;




procedure newest(var nx7: integer;var  xhat7,khat7,h7:
               try; var   y7:real);
      var
         delta: real;
         i,j: integer;
         xsave:try;

         begin
          for i:=1 to nx7 do
             begin
             xsave[i]:=xhat7[i];
             end;
          delta := y7;
              for j:= 1 to nx7 do
                begin
                   delta:= delta - h7[j] * xhat7[j];
                end;
                for i:= 1 to nx7 do
                begin
                   xhat7[i] := xsave[i] + khat7[i] *
                            delta;
                end;
          end;
procedure kgain(nx,ny: integer;var p,q: trytry; r: real;
          var phi,khat: try);


var
    i,j,k    : integer;
    pphi,
    phitp    : matrix20;
    kden     : array20;
```

69

```
begin
    for k:=1 to ny do begin
        kden[k] := 0;
        for i:=1 to nx do begin
            pphi[i,k] := 0;
            phitp[k,i] := 0;
            for j:=1 to nx do begin
                pphi[i,k] := pphi[i,k] + p[i,j]*phi[j];
                phitp[k,i] := phitp[k,i] + phi[j]*p[j,i];
                kden[k] := kden[k] + p[i,j]*phi[j]*phi[i];
            end;
        end;
        kden[k] := kden[k] + r;
    end;
    for k:=1 to ny do begin
        for i:=1 to nx do begin
            for j:=1 to nx do
                p[i,j] := p[i,j] -pphi[i,k]*phitp[k,j]
                    /kden[k] + q[i,j];
            khat[i] := pphi[i,k]/kden[k];
        end;
    end;
end;


                { BEGINNING OF PARAMETER ESTIMATION ROUTINE }


 begin
            if initial = 1
              then
                    begin
                            initializeArrays;
                            goto 3;
                    end;
            ud:=uf;
            filter(up,uf,xu,fc,nf);
            yd := yf;
            filter(yq,yf,xy,fc,nf);
            update(phi,ud,yd,ns);
            phi[nx]:=1.0;
            nsnf := ns + nf;
            time:=time+1;
            if time <= nsnf then goto 3;
            innerprod(yhat,phi,xhat,nx);

        { ESTIMATE BETA TERM IN THETA VECTOR }

            kgain(nx,ny,spt,sq,sr,phi,khat);
            newest(nx,xhat,khat,phi,yf);
            a4:= xhat[1]; a5:= xhat[2]; a6:= xhat[3];
        {COMPUTE BIAS ESTIMATE FROM THE BETA TERM }
```

70

```
            beta[1]:=bias;
            bias:=xhat[3];
            bias:= bias + xhat[1]*beta[1];

        {   BEGINNING OF ESTIMATER FOR PITCH ANGLE }

  f3[1,1]  := 1.0;
  f3[1,2]  := 0.0;
  f3[2,1]  := -vhat/fs;
  f3[2,2]  := 1.0;
  g3[1]    := 1.0/fs;
  g3[2]    := 0.0;
  k[1]     := -12.343/fs;
  k[2]     := 7.2/fs;
  xnew[1] := yf - bias;   {subtract bias from pitchrate}
     for i := 1 to 2 do
       begin
        j := i+1;
        xnew[j] := f3[i,1]*xs[2] + f3[i,2]*xs[3] +
                   g3[i]*xs[1] + k[i]*(z - xs[3]));
     end;
     xs[1] := xnew[1];
     xs[2] := xnew[2];
     xs[3] := xnew[3];
     a1 := xs[1];a2 := xs[2];a3 := xs[3];
3:end;




procedure GenerateDiveplaneCommand (var
depth,auvpitch,auvpitchrate, auvdepthcom,delta:   real;var
k:integer );

{ This procedure takes digitized voltage values of
depth,pitch,pitchrate, and target depth in volys and sends
new commands for control.}

const

{ DT 2801-A DIGITAL TO ANALOG Conversion declarations }

   d2achannel0 = 0;
   pfs = 10.0;
   mfs = - 10.0;
   noc = 4096;
   c1  = 0.5556;
   c2  = 1.0;
   c3  = -0.2143;
   a2  = 0.32;

   scalefact = 2.0;
```

71

```pascal
var
   digitaldatavalue0,digitaldatavalue1, status       :integer;
   e1,e2,e3,vdelta,delta0,sigma                      : real;

function ConvertAnalog2Digital ( analogvalue : real ):
                         integer;

{ This function converts analog signal volts to an
equivalent digital value.  See App D of PCLAB book.}

var
   temp              : real;
begin
    temp :=  ( analogvalue - mfs ) * ( (noc - 10) / (pfs -
          mfs ) );
    convertanalog2digital := round ( temp );

end;


function Sat(sig:real):                          real;
 const
      magmax = 1.0;

    begin
       if (abs(sig) <= magmax) then
        begin
          Sat := sig;
        end
       else if (sig > magmax) then
        begin
          Sat := 1.0;
        end
       else
        begin
         Sat := -1.0;
        end;
    end;

begin  { --------- GenerateDivePlaneCommand ---------- }


        e3:=(auvdepthcom-depth);
        e2:=(auvpitch);
        e1:=(auvpitchrate);

        sigma := (c1*e1)+(c2*e2)+(c3*e3);
        delta0 := -(scalefact)*(sat(sigma));
        delta := delta0 - a2 * e2;
        if k <= 10 then delta:=0.0;
         if(abs(delta)>0.4)  then
          begin
```

72

```
                    delta:=0.4*abs(delta)/delta;
                end;
            vdelta :=-10.0*delta;
            digitaldatavalue0 := convertanalog2digital (vdelta);
            status := dacvalue ( d2achannel0, digitaldatavalue0);


end;    { --------   GenerateDivePlaneCommand ------------ }


procedure InitializeParameters ;

{ This procedure initializes all declared control and
display parameters to zero.}

begin           { ---- procedure InitializeParameters ---    }
        yawrate := 0.0;
        auvdepthvolts := 0.0;
        auvspeedvolts := 0.0;
        auvpitchratevolts := 0.0;
        auvdepth := 0.0;
        auvspeed := 0.0;
        auvpitchrate := 0.0;
        estdepth:=0.0;
        err:=0.0;
        targetdepth := 0.0;
        diveplane := 0.0;
end;            { ---- procedure InitializeParameters ---    }


begin { ----------   ActiveControl   ----------------- }

initializeparameters;
initial := 1;        {initializeArrays in EST procedure}
5:
time := 1;
clrscr;
writeln('DATA FILE NAME? [ie f082201.dat]');
readln(filename);
assign(filevar,filename);
rewrite(filevar);
clrscr;
activemode := run;

repeat {---- Repeat until activemode = exit  ------ }

        { ClosedLoopControlScreen;                          }
        GetTargetDepth ( targetdepth, targetdepthvolts );
        RunModeScreen;
        1:
        while ( not  keyhit ( modereply, modereply2))    do
```

73

```
        begin
          updatecounter := 0;
           while ( updatecounter <  updateincrement ) do
            begin
    GetDigitalSensoryData(auvdepthvolts,yawrate,
                    auvpitchratevolts);

    auvdepth := auvdepthvolts;
    auvpitch  := auvpitchvolts/10.0;
    auvpitchrate := auvpitchratevolts/10.0;

    GenerateDiveplaneCommand ( auvdepth,ac2,ac1,
                  targetdepth,diveplane,time );
     EST (diveplane,auvpitchrate,auvdepth,
         speed,initial,bias1,ac1,ac2,ac3,ac4,ac5,ac6);

writeln(filevar,time:5,auvdepth:12:6,speed:12:6,
      pitchrate:12:6,diveplane:12:6,targetdepth:12:6,
      bias1:12:6,ac1:12:6,ac2:12:6,ac3:12:6,ac4:12:6,
      ac5:12:6,ac6:12:6);

      ATTITUDE_ (targetdepthvolts,auvdepthvolts,attitude);

             initial := 0;
           updatecounter := updatecounter + 1;
           time := time + 1;
          end;    { while updatecounter < updateincrement}


    UpdateRunModeScreen (ac3,ac2,ac1,diveplane,
          targetdepth,activemode,
                    attitude,bias1);


    end;             { while not KeyHit    }

   if (ord(modereply) = 27) and (ord(modereply2) = 59)
   then
        begin
        close(filevar);
         goto 5;
        end
    else if (ord(modereply) = 27) and (ord(modereply2)=61)
      then
        begin
        close(filevar);
        activemode := exit;
      end
until                 (activemode=exit)

end; { --------- ActiveControl    -------------------- }
```

74

```
begin    {----------- StatusAndCommand ------------------- }

   repeat
      StatusAndCommandScreen;
      GetKey ('',mode,mode2);
      if ( ord (mode) = 27 ) and ( ord (mode2) = 59 ) then
         begin
              clrbox2 (x1,y1,x2,y2);
              ClosedLoopControl;
         end;
   until ( mode in ['Q','q'] );

end; {---------- StatusAndCommand ------------------- }


procedure InitializeZeroDigitalSignalOut;

{ This procedure MUST be executed as the first procedure
called in the main program to insure a zero signal out on
the 2 output channels.    Otherwise the DT 2801-A board
defaults to a minimum full scale output.                 }

const

   digitalchan0         = 0;
   digitalchan1         = 1;
   digitalcommandboard = 1;

var
   status,
   digitaldatavalue     : integer;

 begin
   digitaldatavalue := 2048;   { This will be converted
                                 to an equivalent zero analog
                                 signal out on a 12 bit
                                 resolution converter like DT
                                 2801-A.        }
   status := initialize;
   status := selectboard ( digitalcommandboard );
   status := dacvalue ( digitalchan0, digitaldatavalue );
   status := dacvalue ( digitalchan1, digitaldatavalue );
   status := terminate;
end;

procedure DeactivateADBoardAndExitProgram;

{ This procedure deactivates the DT 2801-A board and
presents an exit screen.}
```

```pascal
var
status                  : integer;

begin   { ------ DeactivateADBoardAndExitProgram --------- }
   status := terminate;
   clrbox2 (x1,y1,x2,y2);
   boxprint (y1+10,x1,x2,'THIS CONCLUDES YOUR AUV
AUTOPILOTTING SESSION , BYE');


end;    { ----- DeactivateADBoardAndExitProgram -------- }




BEGIN    { --------------         MAIN PROGRAM     ---------- }

    InitializeZeroDigitalSignalOut;
    clrscr;
    repeat

      MainMenu ( option );

     if ( option in ['R','r']) then
         begin
            repeat
              begin
                StatusAndCommand ( controlmode );
              end;
            until ( controlmode in ['q','Q']);
         end;
    until ( option in ['Q','q']);

    DeactivateADBoardAndExitProgram;

END.  { ----------------- MAIN PROGRAM      ---------- }


III
```

LIST OF REFERENCES

1. Yoerger, Dana R., and Slotine, Jean-Jacques E., <u>Robust Trajectory Control of Underwater Vehicles</u>, IEEE Journal of Oceanographic Engineering, Vol. OE-10, No. 4, pp. 462-470, 1985.

2. Brunner, G.M., <u>Experimental Verification of AUV Performance</u>, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1988.

3. Schwartz, Michael A., <u>Kalman Filtering for Adaptive Depth, Steering, and Roll Control of an Autonomous Underwater Vehicle (AUV)</u>, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1989.

4. Boncal, R.J., <u>A Study of Model Based Maneuvering Controls for Autonomous Underwater Vehicles</u>, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1987.

5. Ljung, Lennart, <u>System Identification Theory for the User</u>, Prentice-Hall, 1987.

6. Friedland, Bernard, <u>Control System Design</u>, McGraw-Hill, 1986.

7. Cristi, R., and Healey, A.J., <u>Adaptive Identification and Control of an AUV</u>, Proc. of 6th International Symposium on Unmanned, Untethered Submersible Technology, Ellicott City, Maryland, June 1989.

8. <u>User Manual</u> for TR-20 Analog Computer, Electronic Associates Inc., West Long Branch, New Jersey, 1968.

9. <u>User Manual</u> for PCLAB, SP041 v2.00, Data Translation, Inc., Marlborough, Massachusetts, 1986.

10. Delaplane, S. W., <u>Preliminary Design and Cycle Verification of a Digital Autopilot for Autonomous Underwater Vehicles</u>, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1988.

INITIAL DISTRIBUTION LIST

No. Copies

1.   Defense Technical Information Center   2
     Carmeron Station
     Alexandria, VA 22304-6145

2.   Library, Code 0142   2
     Naval Postgraduate School
     Monterey, CA 93943-5002

3.   Chairman, Code 62   1
     Department of Electrical and Computer Engineering
     Naval Postgraduate School
     Monterey, CA 93943-5004

4.   Department of Electrical and Computer Engineering   8
     Naval Postgraduate School
     Monterey, CA 93943-5004
     ATTN: Professor R. Cristi, Code 62Cx

5.   Chairman, Department of Mechanical Engineering   2
     Naval Postgraduate School
     Monterey, CA 93943-5004

6.   Head, Undersea AI and Robotics Branch   1
     Naval Ocean System Center
     San Diego, CA 92152
     ATTN: P. Heckman, Code 943

7.   Commander, Naval Sea Systems Command (PMS-350)   1
     Washington, DC 20362-5101
     ATTN: RADM Evans, Code SEA92R

8.   Commander, Naval Coastal Systems Center   1
     Panama City, FL 32407-5000
     ATTN: Dr. G. Dobeck

9.   Dr. D. Milne, Code 1563   1
     DTRC, Carderock
     Bethesda, MD 20084-5000

10. Commander, Naval Sea Systems Command   1
     Robotics Office
     Washington, DC 20362-5101
     ATTN: LT Relle Lyman

11. Commander, Naval Research Laboratory   1
    Washington, DC   20362-5101
    ATTN: D. Steiger

12. Commander, Naval Surface Weapons Center   1
    White Oak, MD   20910
    ATTN: H. Cook, Code U25

13. Distinguished Professor G. Thaler, Code 62Tr   1
    Electrical and Computer Engineering Department
    Naval Postgraduate School
    Monterey, CA 93943-5004

14. LT James M. Williams   1
    9895 Scripps Westview Way #148
    San Diego, CA 92131